OpenShift Container Platform 4.16

インストール

OpenShift Container Platform クラスターのインストールおよび設定

Last Updated: 2024-06-26

OpenShift Container Platform 4.16 インストール

OpenShift Container Platform クラスターのインストールおよび設定

法律上の通知

概要

このドキュメントでは、OpenShift Container Platform のインストール方法と、一部の設定プロセスの詳細を説明します。

目次

第1章 OPENSHIFT CONTAINER PLATFORM インストールの概要

1.1. OPENSHIFT CONTAINER PLATFORM のインストール

OpenShift Container Platform インストールプログラムでは、以下に詳細がリストされている 4 つの方法でクラスターをデプロイできます。

- **インタラクティブ**: Web ベースの Assisted Installer を使用してクラスターをデプロイできます。これは、ネットワークがインターネットに接続されているクラスターに最適です。Assisted Installer は、OpenShift Container Platform をインストールする最も簡単な方法であり、スマートなデフォルトを提供し、クラスターをインストールする前に事前検証を実行します。また、自動化および高度な設定シナリオのための RESTful API も提供します。

- **ローカルエージェントベース**: 非接続環境またはネットワークが制限された環境では、Agent-based Installer を使用してクラスターをローカルにデプロイできます。この方法では、Assisted Installer の多くの利点を得られますが、最初に Agent-based Installer をダウンロードして設定する必要があります。設定はコマンドラインインターフェイスで行います。このアプローチは、非接続環境に最適です。

- **自動化**: installer-provisioned infrastructure にクラスターをデプロイできます。インストールプログラムは、各クラスターホストのベースボード管理コントローラー (BMC) をプロビジョニングに使用します。接続環境または非接続環境でクラスターをデプロイできます。

- **完全な制御**: お客様が準備および保守するインフラストラクチャーにクラスターをデプロイメントできます。これにより、最大限のカスタマイズ性が提供されます。接続環境または非接続環境でクラスターをデプロイできます。

それぞれの方法でデプロイしたクラスターは、以下の特性を持ちます。

- 単一障害点のない高可用性インフラストラクチャーがデフォルトで利用可能です。

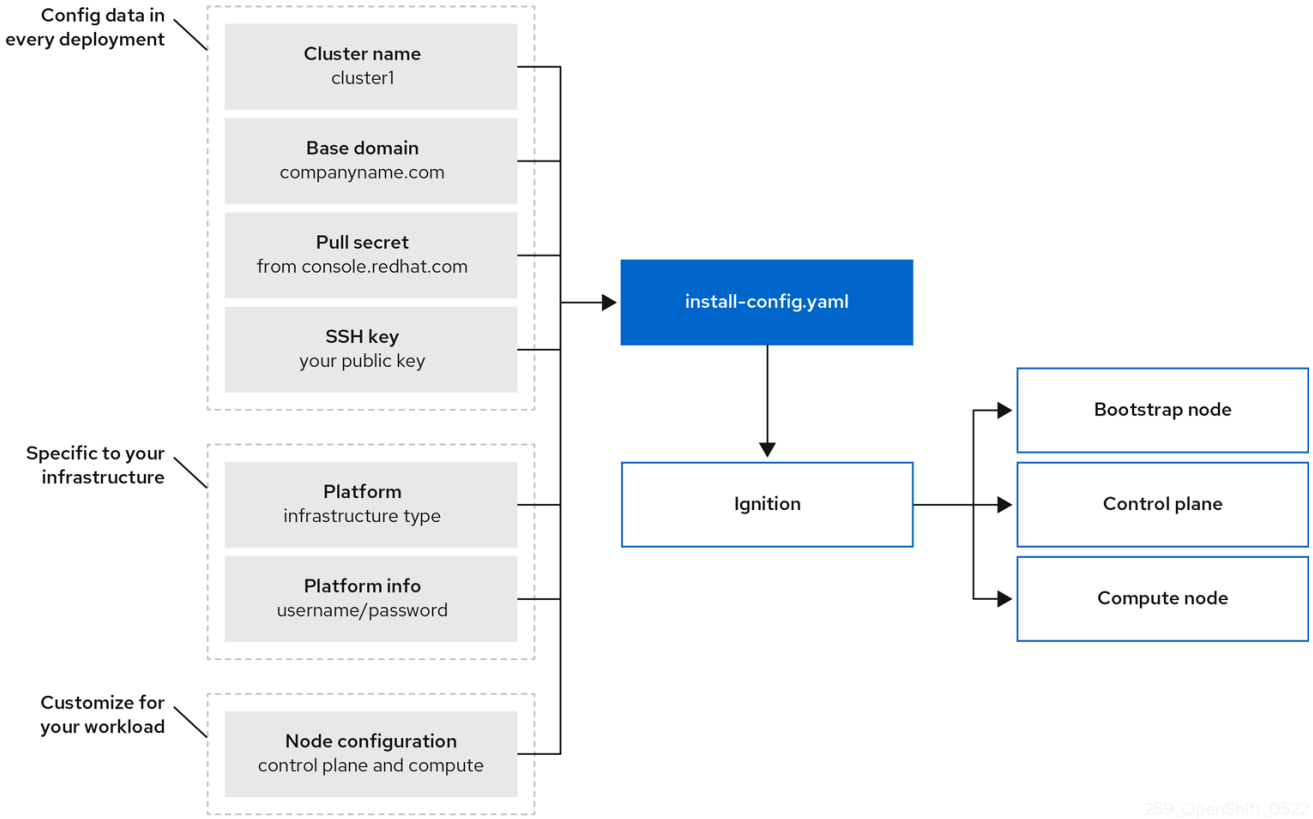
- 管理者は適用される更新の内容とタイミングを制御できます。

1.1.1. インストールプログラムについて

インストールプログラムを使用して、各タイプのクラスターをデプロイメントできます。インストールプログラムは、ブートストラップ、コントロールプレーン、コンピュートマシンの Ignition 設定ファイルなどのメインアセットを生成します。インフラストラクチャーを適切に設定している場合、これらの 3 つのマシン設定を使用して OpenShift Container Platform クラスターを起動できます。

OpenShift Container Platform インストールプログラムは、クラスターのインストールを管理するために一連のターゲットおよび依存関係を使用します。インストールプログラムには、達成する必要のある一連のターゲットが設定され、それぞれのターゲットには一連の依存関係が含まれます。各ターゲットはそれぞれの依存関係の条件が満たされ次第、別個に解決されるため、インストールプログラムは複数のターゲットを並行して達成できるように動作し、最終的にクラスターが実行するようにします。プログラムが依存関係を満たしているため、インストールプログラムはコマンドを実行してコンポーネントを再作成する代わりに、既存のコンポーネントを認識して使用します。

図1.1 OpenShift Container Platform インストールのターゲットおよび依存関係



1.1.2. Red Hat Enterprise Linux CoreOS (RHCOS) について

インストール後に、各クラスターマシンは Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングマシンとして使用します。RHCOS は Red Hat Enterprise Linux (RHEL) の不変のコンテナーホストのバージョンであり、デフォルトで SELinux が有効になった RHEL カーネルを特長としています。RHCOS には、Kubernetes ノードエージェントである **kubelet** や、Kubernetes に対して最適化される CRI-O コンテナーランタイムが含まれます。

OpenShift Container Platform 4.16 クラスターのすべてのコントロールプレーンは、Ignition と呼ばれる最初の起動時に使用される重要なプロビジョニングツールが含まれる RHCOS を使用する必要があります。このツールは、クラスターのマシンの設定を可能にします。オペレーティングシステムの更新は、**OSTree** をバックエンドとして使用する起動可能なコンテナーイメージとして配信され、Machine Config Operator によりクラスター全体にデプロイされます。実際のオペレーティングシステムの変更は、**rpm-ostree** を使用することにより、atomic 操作として各マシン上でインプレースで行われます。これらのテクノロジーを組み合わせることで、OpenShift Container Platform は、プラットフォーム全体を最新の状態に保つインプレースアップグレードによって、クラスター上の他のアプリケーションを管理するのと同じようにオペレーティングシステムを管理できるようになります。これらのインプレースアップグレードにより、オペレーションチームの負担を軽減できます。

すべてのクラスターマシンのオペレーティングシステムとして RHCOS を使用する場合、クラスターはオペレーティングシステムを含むコンポーネントとマシンのあらゆる側面を管理します。このため、マシンを変更できるのは、インストールプログラムと Machine Config Operator だけです。インストールプログラムは、Ignition 設定ファイルを使用して各マシンの正確な状態を設定し、インストール後に Machine Config Operator が新しい証明書やキーの適用などのマシンへの追加の変更を完了します。

1.1.3. OpenShift Container Platform のインストールに関する一般的な用語集

この用語集では、インストールコンテンツに関する一般的な用語を定義しています。インストールプロセスの理解を深めるために、次の用語リストを確認してください。

Assisted Installer

クラスター設定を作成するための Web ベースのユーザーインターフェイスまたは RESTful API を提供する、console.redhat.com でホストされるインストーラー。Assisted Installer は検出イメージを生成します。クラスターマシンは、RHCOS とエージェントをインストールする検出イメージで起動します。Assisted Installer とエージェントは、ともにインストール前の検証とクラスターのインストールを提供します。

Agent-based Installer

Assisted Installer に似たインストーラーですが、最初に Agent-based Installer をダウンロードする必要があります。Agent-based Installer は非接続環境に最適です。

ブートストラップノード

OpenShift Container Platform コントロールプレーンをデプロイするために必要な最小限の Kubernetes 設定を実行する一時的なマシン。

コントロールプレーン

コンテナーのライフサイクルを定義、デプロイ、および管理するための API とインターフェイスを公開するコンテナーオーケストレーションレイヤー。コントロールプレーンマシンとも呼ばれます。

コンピュートノード

クラスターユーザーのワークロードを実行するノード。ワーカーノードとしても知られています。

非接続インストール

場合によっては、プロキシーサーバーを介しても、データセンターの一部はインターネットにアクセスできない可能性があります。このような環境でも OpenShift Container Platform をインストールできますが、必要なソフトウェアおよびイメージをダウンロードし、これらを非接続環境で利用できる状態にする必要があります。

OpenShift Container Platform インストールプログラム

インフラストラクチャーをプロビジョニングし、クラスターをデプロイするプログラム。

installer-provisioned infrastructure

インストールプログラムは、クラスターを実行するインフラストラクチャーをデプロイして設定します。

Ignition 設定ファイル

オペレーティングシステムの初期化中に Ignition ツールが Red Hat Enterprise Linux CoreOS (RHCOS) を設定するために使用するファイル。インストールプログラムは、ブートストラップ、コントロールプレーン、およびワーカーノードを初期化するために、さまざまな Ignition 設定ファイルを生成します。

Kubernetes マニフェスト

JSON または YAML 形式の Kubernetes API オブジェクトの仕様。設定ファイルには、デプロイメント、設定マップ、シークレット、デーモンセットなどを含めることができます。

Kubelet

コンテナーが Pod で実行されていることを確認するために、クラスター内の各ノードで実行されるプライマリーノードエージェント。

ロードバランサー

ロードバランサーは、クライアントに対する単一の通信先として機能します。API のロードバランサーは、着信トラフィックをコントロールプレーンノード全体に分散します。

Machine Config Operator

クラスター内のノードのカーネルと kubelet との間にあるすべてのものを含む、基本オペレーティングシステムとコンテナーランタイムの設定と更新を管理および適用する Operator。

Operator

OpenShift Container Platform クラスターで Kubernetes アプリケーションをパッケージ化、デプロイ、および管理するための推奨される方法。Operator は、人間の操作に関する知識を取り入れて、簡単にパッケージ化してお客様と共有できるソフトウェアにエンコードします。

user-provisioned infrastructure

OpenShift Container Platform は、ユーザーが独自にプロビジョニングするインフラストラクチャーにインストールできます。インストールプログラムを使用すると、クラスターインフラストラクチャーのプロビジョニングに必要なアセットを生成し、クラスターインフラストラクチャーを作成して、提供したインフラストラクチャーにクラスターをデプロイできます。

1.1.4. インストールプロセス

Assisted Installer を除き、OpenShift Container Platform クラスターをインストールする場合は、OpenShift Cluster Manager Hybrid Cloud Console の適切な **クラスタータイプ** ページから、インストールプログラムをダウンロードする必要があります。このコンソールは以下を管理します。

- アカウントの REST API。
- 必要なコンポーネントを取得するために使用するプルシークレットであるレジストリートークン。
- クラスターのアイデンティティーを Red Hat アカウントに関連付けて使用状況のメトリクスの収集を容易にするクラスター登録。

OpenShift Container Platform 4.16 では、インストールプログラムは、一連のアセットに対して一連のファイル変換を実行する Go バイナリーファイルです。インストールプログラムと対話する方法は、インストールタイプによって異なります。次のインストールユースケースを検討してください。

- Assisted Installer を使用してクラスターをデプロイするには、Assisted Installer を使用してクラスター設定を行う必要があります。ダウンロードして設定するインストールプログラムはありません。クラスター設定が完了したら、検出 ISO をダウンロードし、そのイメージを使用してクラスターマシンを起動します。Assisted Installer を使用して、完全に統合された Nutanix、vSphere、およびベアメタル、ならびに統合されていないその他のプラットフォームにクラスターをインストールできます。ベアメタルにインストールする場合は、ネットワーク、負荷分散、ストレージ、個々のクラスターマシンなど、すべてのクラスターインフラストラクチャーとリソースを提供する必要があります。
- Agent-based Installer を使用してクラスターをデプロイするには、最初に Agent-based Installer をダウンロードします。次に、クラスターを設定して、検出イメージを生成します。検出イメージを使用してクラスターマシンを起動します。これにより、インストールプログラムと通信してプロビジョニングを処理するエージェントがインストールされます。インストールプログラムとを操作したりプロビジョナーマシンを自分で設定したりする必要はありません。ネットワーク、負荷分散、ストレージ、個々のクラスターマシンなど、すべてのクラスターインフラストラクチャーとリソースを提供する必要があります。このアプローチは、非接続環境に最適です。
- インストーラーでプロビジョニングされるインフラストラクチャーのクラスターの場合、インフラストラクチャーのブートストラップおよびプロビジョニングは、ユーザーが独自に行うのではなくインストールプログラムが代行します。インストールプログラムは、ベアメタルにインストールする場合を除き、クラスターをサポートするために必要なすべてのネットワーク、マシン、およびオペレーティングシステムを作成します。ベアメタルにインストールする場合は、ブートストラップマシン、ネットワーク、負荷分散、ストレージ、個々のクラスターマシンなど、すべてのクラスターインフラストラクチャーとリソースを提供する必要があります。
- クラスターのインフラストラクチャーを独自にプロビジョニングし、管理する場合には、ブートストラップマシン、ネットワーク、負荷分散、ストレージ、および個々のクラスターマシンを含む、すべてのクラスターインフラストラクチャーおよびリソースを指定する必要がありま

す。

インストールプログラムの場合、プログラムはインストール中に 3 つのファイルセットを使用します。それは、**install-config.yaml** という名前のインストール設定ファイル、Kubernetes マニフェスト、およびマシンタイプの Ignition 設定ファイルです。

重要

インストール時に、Kubernetes および基礎となる RHCOS オペレーティングシステムを制御する Ignition 設定ファイルを変更できます。ただし、これらのオブジェクトに対して加える変更の適合性を確認するための検証の方法はなく、これらのオブジェクトを変更するとクラスターが機能しなくなる可能性があります。これらのオブジェクトを変更する場合、クラスターが機能しなくなる可能性があります。このリスクがあるために、変更方法についての文書化された手順に従っているか、Red Hat サポートが変更することを指示した場合を除き、Kubernetes および Ignition 設定ファイルの変更はサポートされていません。

インストール設定ファイルは Kubernetes マニフェストに変換され、その後マニフェストは Ignition 設定にラップされます。インストールプログラムはこれらの Ignition 設定ファイルを使用してクラスターを作成します。

インストール設定ファイルはインストールプログラムの実行時にすべてプルーニングされるため、再び使用する必要のあるすべての設定ファイルをバックアップしてください。

重要

インストール時に設定したパラメーターを変更することはできませんが、インストール後に数多くのクラスター属性を変更することができます。

Assisted Installer を使用したインストールプロセス
Assisted Installer を使用したインストールでは、Web ベースのユーザーインターフェイスまたは RESTful API を使用して対話的にクラスター設定を作成します。Assisted Installer ユーザーインターフェイスは、ユーザーインターフェイスまたは API で変更しない限り、必要な値の入力を求め、残りのパラメーターに適切なデフォルト値を提供します。Assisted Installer は検出イメージを生成します。このイメージをダウンロードして、クラスターマシンの起動に使用します。イメージによって RHCOS とエージェントがインストールされ、エージェントがプロビジョニングを処理します。OpenShift Container Platform を、Assisted Installer を使用して完全な統合により、Nutanix、vSphere、およびベアメタルにインストールできます。統合せずに、Assisted Installer を使用して OpenShift Container Platform を別のプラットフォームにインストールすることも可能です。

OpenShift Container Platform は、オペレーティングシステム自体を含む、クラスターのすべての側面を管理します。各マシンは、それが参加するクラスターでホストされるリソースを参照する設定に基づいて起動します。この設定により、クラスターは更新の適用時に自己管理できます。

可能であれば、Agent-based Installer をダウンロードして設定する必要がないように、Assisted Installer 機能を使用してください。

エージェントベースのインフラストラクチャーを使用したインストールプロセス
エージェントベースのインストールは Assisted Installer を使用する場合とよく似ていますが、最初にエージェントベースのインストーラ をダウンロードしてインストールする必要があります。エージェントベースのインストールは、Assisted Installer の利便性を活用したいが、非接続環境でクラスターをインストールする必要がある場合に役立ちます。

可能であれば、エージェントベースのインストール機能を使用してください。その場合、ブートストラップ仮想マシンを使用してプロビジョナーマシンを作成し、クラスターインフラストラクチャーをプロビジョニングして維持する必要がなくなります。

インストーラーでプロビジョニングされるインフラストラクチャーでのインストールプロセス
デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーです。デフォルトで、インストールプログラムはインストールウィザードとして機能し、独自に判別できない値の入力を求めるプロンプトを出し、残りのパラメーターに妥当なデフォルト値を提供します。インストールプロセスは、高度なインフラストラクチャーシナリオに対応するようカスタマイズすることもできます。インストールプログラムは、クラスターの基盤となるインフラストラクチャーをプロビジョニングします。

標準クラスターまたはカスタマイズされたクラスターのいずれかをインストールすることができます。標準クラスターの場合、クラスターをインストールするために必要な最小限の詳細情報を指定します。カスタマイズされたクラスターの場合、コントロールプレーンが使用するマシン数、クラスターがデプロイする仮想マシンのタイプ、または Kubernetes サービスネットワークの CIDR 範囲などのプラットフォームについての詳細を指定することができます。

可能な場合は、この機能を使用してクラスターインフラストラクチャーのプロビジョニングと保守の手間を省くようにしてください。他のすべての環境の場合には、インストールプログラムを使用してクラスターインフラストラクチャーをプロビジョニングするために必要なアセットを生成できます。

インストーラーでプロビジョニングされるインフラストラクチャークラスターの場合、OpenShift Container Platform は、オペレーティングシステム自体を含むクラスターのすべての側面を管理します。各マシンは、それが参加するクラスターでホストされるリソースを参照する設定に基づいて起動します。この設定により、クラスターは更新の適用時に自己管理できます。

ユーザーによってプロビジョニングされるインフラストラクチャーを使用したインストールプロセス
OpenShift Container Platform はユーザーが独自にプロビジョニングするインフラストラクチャーにインストールすることもできます。インストールプログラムを使用してクラスターインフラストラクチャーのプロビジョニングに必要なアセットを生成し、クラスターインフラストラクチャーを作成し、その後にクラスターをプロビジョニングしたインフラストラクチャーにデプロイします。

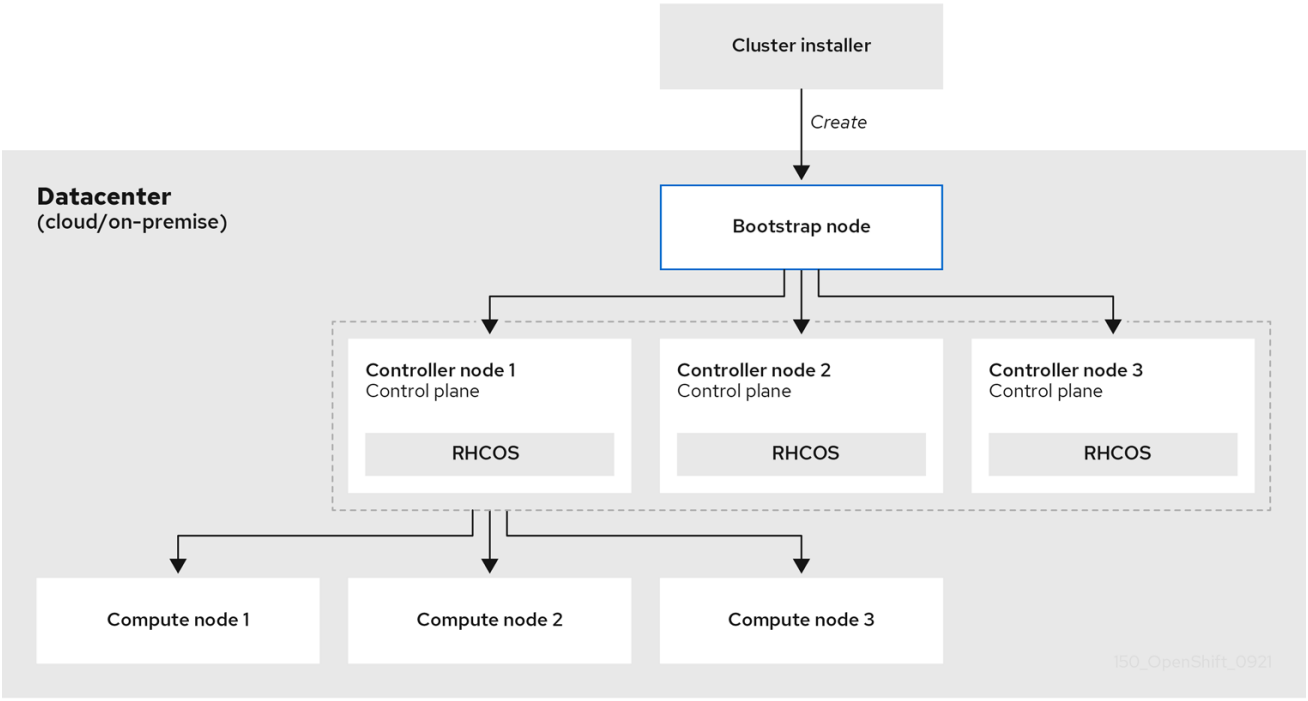
インストールプログラムがプロビジョニングしたインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。次のリストは、一部のセルフマネージドリソースの詳細を示しています。

- クラスターを設定するコントロールプレーンおよびコンピュートマシンの基礎となるインフラストラクチャー
- ロードバランサー
- DNS レコードおよび必要なサブネットを含むクラスターネットワーク
- クラスターインフラストラクチャーおよびアプリケーションのストレージ

クラスターでユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合には、RHEL コンピュートマシンをクラスターに追加するオプションを使用できます。

インストールプロセスの詳細
クラスターがプロビジョニングされると、クラスター内の各マシンにはクラスターに関する情報が必要になります。OpenShift Container Platform は初期設定時に一時的なブートストラップマシンを使用して、必要な情報を永続的なコントロールプレーンに提供します。一時的なブートストラップマシンは、クラスターの作成方法を記述する Ignition 設定ファイルを使用して起動します。ブートストラップマシンは、コントロールプレーンを設定するコントロールプレーンマシンを作成します。その後、コントロールプレーンマシンはコンピュートマシン (ワーカーマシンとしても知られる) を作成します。以下の図はこのプロセスを示しています。

図1.2 ブートストラップ、コントロールプレーンおよびコンピュートマシンの作成



クラスターマシンを初期化した後、ブートストラップマシンは破棄されます。すべてのクラスターがこのブートストラッププロセスを使用してクラスターを初期化しますが、ユーザーがクラスターのインフラストラクチャーをプロビジョニングする場合には、多くの手順を手動で実行する必要があります。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間でローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを検討してください。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

クラスターのブートストラップには、以下のステップが関係します。

1. ブートストラップマシンが起動し、コントロールプレーンマシンの起動に必要なリモートリソースのホスティングを開始します。インフラストラクチャーをプロビジョニングする場合、この手順では人的介入が必要になります。
2. ブートストラップマシンは、単一ノードの etcd クラスターと一時的な Kubernetes コントロールプレーンを起動します。

3. コントロールプレーンマシンは、ブートストラップマシンからリモートリソースをフェッチし、起動を終了します。インフラストラクチャーをプロビジョニングする場合、この手順では人的介入が必要になります。
4. 一時的なコントロールプレーンは、実稼働コントロールプレーンマシンに対して実稼働コントロールプレーンをスケジュールします。
5. Cluster Version Operator (CVO) はオンラインになり、etcd Operator をインストールします。etcd Operator はすべてのコントロールプレーンノードで etcd をスケールアップします。
6. 一時的なコントロールプレーンはシャットダウンし、コントロールを実稼働コントロールプレーンに渡します。
7. ブートストラップマシンは OpenShift Container Platform コンポーネントを実稼働コントロールプレーンに挿入します。
8. インストールプログラムはブートストラップマシンをシャットダウンします。インフラストラクチャーをプロビジョニングする場合、この手順では人的介入が必要になります。
9. コントロールプレーンはコンピュートノードを設定します。
10. コントロールプレーンは一連の Operator の形式で追加のサービスをインストールします。

このブートストラッププロセスの結果として、OpenShift Container Platform クラスターが実行されます。次に、クラスターはサポートされる環境でのコンピュートマシンの作成など、日常の操作に必要な残りのコンポーネントをダウンロードし、設定します。

関連情報

- Red Hat OpenShift Network Calculator

1.1.5. インストール後のノード状態の確認

以下のインストールヘルスチェックが正常に行われると、OpenShift Container Platform のインストールが完了します。

- プロビジョナーは、OpenShift Container Platform Web コンソールにアクセスできます。
- すべてのコントロールプレーンノードが準備状態にある。
- すべてのクラスター Operator が利用可能です。

注記

インストールが完了すると、ワーカーノードを実行する特定のクラスター Operator が継続的にすべてのワーカーノードのプロビジョニングを試みます。すべてのワーカーノードが **READY** と報告されるまで、多少時間がかかります。ベアメタルへのインストールの場合、ワーカーノードのトラブルシューティングを行う前に、少なくとも 60 分間待機してください。他のすべてのプラットフォームへのインストールの場合は、ワーカーノードのトラブルシューティングを行う前に、少なくとも 40 分間待機してください。ワーカーノードを実行するクラスター Operator の **DEGRADED** 状態は、ノードの状態ではなく、Operator 自体のリソースに依存します。

インストールが完了したら、引き続きクラスター内におけるノードの状態を監視できます。

前提条件

- インストールプログラムはターミナルで正常に解決されます。

手順

1. すべてのワーカーノードのステータスを表示します。

```
$ oc get nodes
```

出力例

```
NAME                         STATUS   ROLES    AGE   VERSION
example-compute1.example.com   Ready    worker   13m   v1.21.6+bb8d50a
example-compute2.example.com   Ready    worker   13m   v1.21.6+bb8d50a
example-compute4.example.com   Ready    worker   14m   v1.21.6+bb8d50a
example-control1.example.com   Ready    master   52m   v1.21.6+bb8d50a
example-control2.example.com   Ready    master   55m   v1.21.6+bb8d50a
example-control3.example.com   Ready    master   55m   v1.21.6+bb8d50a
```

2. すべてのワーカーマシンノードのフェーズを表示します。

```
$ oc get machines -A
```

出力例

```
NAMESPACE               NAME                              PHASE        TYPE   REGION   ZONE   AGE
openshift-machine-api   example-zbbt6-master-0            Running                             95m
openshift-machine-api   example-zbbt6-master-1            Running                             95m
openshift-machine-api   example-zbbt6-master-2            Running                             95m
openshift-machine-api   example-zbbt6-worker-0-25bhp      Running                             49m
openshift-machine-api   example-zbbt6-worker-0-8b4c2      Running                             49m
openshift-machine-api   example-zbbt6-worker-0-jkbqt      Running                             49m
openshift-machine-api   example-zbbt6-worker-0-qrl5b      Running                             49m
```

関連情報

- BareMetalHost リソースの取得
- インストール後
- インストールの検証
- Agent-based Installer
- OpenShift Container Platform のアシステッドインストーラー

インストールのスコープ

OpenShift Container Platform インストールプログラムのスコープは意図的に狭められています。単純さを確保し、確実にインストールを実行できるように設計されているためです。インストールが完了した後に数多くの設定タスクを実行することができます。

関連情報

- OpenShift Container Platform 設定リソースの詳細は、利用可能なクラスターのカスタマイズ を参照してください。

1.1.6. OpenShift Local の概要

OpenShift Local は、OpenShift Container Platform クラスターのビルドを開始するための迅速なアプリケーション開発をサポートします。OpenShift Local は、ローカルのコンピューターで実行し、セットアップおよびテストをシンプル化し、コンテナーベースのアプリケーションを開発するのに必要なすべてのツールと共にクラウド開発環境をローカルにエミュレートすることを目的として設計されています。

OpenShift Local は、使用するプログラミング言語にかかわらずアプリケーションをホストし、事前に設定された最小限の Red Hat OpenShift Container Platform クラスターをローカル PC に提供します。その際に、サーバーベースのインフラストラクチャーは必要ありません。

ホストされる環境では、OpenShift Local はマイクロサービスを作成してイメージに変換し、Linux、macOS、または Windows 10 以降を実行するノートパソコンまたはデスクトップ上の Kubernetes がホストするコンテナーで直接それらを実行できます。

OpenShift Local の詳細は、Red Hat OpenShift Local の概要 を参照してください。

1.2. OPENSHIFT CONTAINER PLATFORM クラスターでサポートされるプラットフォーム

OpenShift Container Platform バージョン 4.16 では、installer-provisioned infrastructure を使用するクラスターの場合、以下のプラットフォームにインストールできます。

- Alibaba Cloud
- Amazon Web Services (AWS)
- ベアメタル
- Google Cloud Platform (GCP)
- IBM Cloud®
- Microsoft Azure
- Microsoft Azure Stack Hub
- Nutanix
- Red Hat OpenStack Platform (RHOSP)
    - OpenShift Container Platform の最新リリースは、最新の RHOSP のロングライフリリースおよび中間リリースの両方をサポートします。RHOSP リリースの互換性についての詳細は、OpenShift Container Platform on RHOSP support matrix を参照してください。
- VMware vSphere

これらのクラスターの場合、インストールプロセスを実行するコンピューターを含むすべてのマシンが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供できるようインターネットに直接アクセスできる必要があります。

重要

インストール後は、以下の変更はサポートされません。

- クラウドプロバイダープラットフォームの混在。
- クラウドプロバイダーコンポーネントの混在。たとえば、クラスターをインストールしたプラットフォーム上の別のプラットフォームから永続ストレージフレームワークを使用します。

OpenShift Container Platform 4.16 では、user-provisioned infrastructure を使用するクラスターの場合、以下のプラットフォームにインストールできます。

- AWS
- Azure
- Azure Stack Hub
- ベアメタル
- GCP
- IBM Power®
- IBM Z® または IBM® LinuxONE
- RHOSP
  - OpenShift Container Platform の最新リリースは、最新の RHOSP のロングライフリリースおよび中間リリースの両方をサポートします。RHOSP リリースの互換性についての詳細は、OpenShift Container Platform on RHOSP support matrix を参照してください。
- VMware Cloud on AWS
- VMware vSphere

プラットフォームでサポートされているケースに応じて、user-provisioned infrastructure でインストールを実行できます。これにより、完全なインターネットアクセスでのマシンの実行、プロキシーの背後へのクラスターの配置、非接続インストールの実行が可能になります。

非接続インストールでは、クラスターのインストールに必要なイメージをダウンロードして、ミラーレジストリーに配置し、そのデータを使用してクラスターをインストールできます。vSphere またはベアメタルインフラストラクチャー上での非接続インストールでは、プラットフォームコンテナーのイメージをプルするためにインターネットにアクセスする必要がありますが、クラスターマシンはインターネットへの直接のアクセスを必要としません。

OpenShift Container Platform 4.x Tested Integrations のページには、各種プラットフォームの統合テストについての詳細が記載されています。

関連情報

- それぞれのサポートされているプラットフォームで利用できるインストールのタイプについての詳細は、各種プラットフォームのサポートされているインストール方法 を参照してください。

- インストール方法の選択と必要なリソースの準備については、クラスターインストール方法の選択およびそのユーザー向けの準備 を参照してください。

- Red Hat OpenShift Network Calculator は、デプロイフェーズと拡張フェーズの両方で、クラスターネットワークを設計するのに役立ちます。クラスターネットワークに関連する一般的な疑問を解消し、便利な JSON 形式で出力を提供します。

第2章 クラスターインストール方法の選択およびそのユーザー向けの準備

OpenShift Container Platform をインストールする前に、実行するインストールプロセスを決定し、ユーザー用にクラスターを準備する際に必要なすべてのリソースがあることを確認します。

2.1. クラスターのインストールタイプの選択

OpenShift Container Platform クラスターをインストールする前に、実行する最適なインストール手順を選択する必要があります。以下の質問に回答して、最も良いオプションを選択します。

2.1.1. OpenShift Container Platform クラスターを独自にインストールし、管理しますか？

OpenShift Container Platform を独自にインストールし、管理する必要がある場合、以下のプラットフォームにインストールすることができます。

- 64 ビット x86 インスタンスの Amazon Web Services (AWS)
- 64 ビット ARM インスタンスの Amazon Web Services (AWS)
- 64 ビット x86 インスタンスの Microsoft Azure
- 64 ビット ARM インスタンスの Microsoft Azure
- Microsoft Azure Stack Hub
- 64 ビット x86 インスタンス上の Google Cloud Platform (GCP)
- 64 ビット ARM インスタンス上の Google Cloud Platform (GCP)
- Red Hat OpenStack Platform (RHOSP)
- IBM Cloud®
- IBM Z® または IBM® LinuxONE
- IBM Z® または IBM® LinuxONE for Red Hat Enterprise Linux (RHEL) KVM
- IBM Power®
- IBM Power® Virtual Server
- Nutanix
- VMware vSphere
- ベアメタルまたはその他のプラットフォームに依存しないインフラストラクチャー

OpenShift Container Platform 4 クラスターは、オンプレミスハードウェアとクラウドホストサービスの両方にデプロイできますが、クラスターのすべてのマシンは同じデータセンターまたはクラウドホストサービスにある必要があります。

OpenShift Container Platform を使用する必要があるが、クラスターを独自に管理することを望まない場合は、複数のマネージドサービスオプションを使用できます。Red Hat によって完全に管理されるク

ラスターが必要な場合は、OpenShift Dedicated または OpenShift Online を使用することができます。OpenShift を Azure、AWS、IBM Cloud®、または Google Cloud でマネージドサービスとして使用することもできます。マネージドサービスの詳細は、OpenShift の製品 ページを参照してください。クラウド仮想マシンを仮想ベアメタルとして OpenShift Container Platform クラスターをインストールする場合、対応するクラウドベースのストレージはサポートされません。

2.1.2. OpenShift Container Platform 3 を使用したことがあり、その上で OpenShift Container Platform 4 を使用することを希望していますか？

OpenShift Container Platform 3 を使用したことがあり、OpenShift Container Platform 4 を使用してみたいと思われる場合は、OpenShift Container Platform 4 がどのように異なるかを理解しておく必要があります。OpenShift Container Platform 4 では、 Kubernetes アプリケーション、プラットフォームが実行されるオペレーティングシステム、Red Hat Enterprise Linux CoreOS (RHCOS) を共にシームレスにパッケージ化し、デプロイし、管理する Operator を使用します。マシンをデプロイし、それらのオペレーティングシステムを設定して OpenShift Container Platform をそれらにインストールできるようにする代わりに、RHCOS オペレーティングシステムが OpenShift Container Platform クラスターの統合された部分として使用されます。OpenShift Container Platform のインストールプロセスの一部として、クラスターマシンのオペレーティングシステムをデプロイします。OpenShift Container Platform 3 と 4 の相違点 を参照してください。

OpenShift Container Platform クラスターのインストールプロセスの一部としてマシンをプロビジョニングする必要があるため、OpenShift Container Platform 3 クラスターを OpenShift Container Platform 4 にアップグレードすることはできません。その代わりに、新規の OpenShift Container Platform 4 クラスターを作成し、OpenShift Container Platform 3 ワークロードをそれらに移行する必要があります。移行の詳細は、OpenShift Container Platform 3 から 4 への移行の概要 を参照してください。OpenShift Container Platform 4 に移行するにあたり、任意のタイプの実稼働用のクラスターのインストールプロセスを使用して新規クラスターを作成できます。

2.1.3. クラスターで既存のコンポーネントを使用する必要がありますか？

オペレーティングシステムは OpenShift Container Platform に不可欠な要素であり、OpenShift Container Platform のインストールプログラムはすべてのインフラストラクチャーの起動を簡単に実行できます。これらは、**インストーラーでプロビジョニングされるインフラストラクチャー** のインストールと呼ばれています。この種のインストールでは、ユーザーは既存のインフラストラクチャーをクラスターに提供できますが、インストールプログラムがクラスターを最初に必要とするすべてのマシンをデプロイします。

クラスターまたはその基盤となるマシンの Alibaba Cloud、AWS、Azure、Azure Stack Hub、GCP、または Nutanix へのカスタマイズを指定することなく、インストーラーでプロビジョニングされたインフラストラクチャークラスターをデプロイできます。

クラスターマシンのインスタンスタイプなど、インストーラーでプロビジョニングされるインフラストラクチャークラスターの基本設定を実行する必要がある場合は、AWS、Azure、または GCP のインストールをカスタマイズできます。

インストーラーでプロビジョニングされるインフラストラクチャーのインストールの場合、AWS の既存の VPC、Azure の vNet、または GCP の VPC を使用できます。ネットワークインフラストラクチャーの一部を再利用して、AWS、Azure、または GCP のクラスターが環境内の既存の IP アドレスの割り当てと共存し、既存の MTU および VXLAN 設定と統合するようにできます。これらのクラウドに既存のアカウントおよび認証情報がある場合は、それらを再利用できますが、OpenShift Container Platform クラスターをインストールするために必要なパーミッションを持つようアカウントを変更する必要がある場合があります。

installer-provisioned infrastructure メソッドを使用して、vSphere および ベアメタル のハードウェアに適切なマシンインスタンスを作成できます。さらに、vSphere では、インストール時に追加のネットワークパラメーターをカスタマイズすることもできます。

大規模なクラウドインフラストラクチャーを再利用する必要がある場合、**ユーザーによってプロビジョニングされるインフラストラクチャー** のインストールを実行できます。これらのインストールでは、インストールプロセス時にクラスターに必要なマシンを手動でデプロイします。AWS、Azure、および Azure Stack Hub でユーザーによってプロビジョニングされるインフラストラクチャーを実行する場合、提供されるテンプレートを使用して必要なすべてのコンポーネントを起動できます。共有 VPC on GCP を再利用することもできます。それ以外の場合は、プロバイダーに依存しない インストール方法を使用して、クラスターを他のクラウドにデプロイすることができます。

ユーザーによってプロビジョニングされるインフラストラクチャーは、既存のハードウェアで実行することもできます。RHOSP、IBM Z® または IBM® LinuxONE、RHEL KVM が搭載された IBM Z® および IBM® LinuxONE、IBM Power、または vSphere を使用する場合は、特定のインストール手順に従ってクラスターをデプロイします。サポートされる他のハードウェアを使用する場合は、ベアメタルのインストール 手順に従います。vSphere や ベアメタル などの一部のプラットフォームの場合は、インストール時に追加のネットワークパラメーターをカスタマイズすることもできます。

2.1.4. クラスターに追加のセキュリティーが必要ですか？

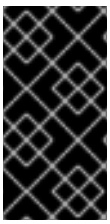
ユーザーによってプロビジョニングされるインストール方法を使用する場合、クラスターのプロキシーを設定できます。この手順は各インストール手順に含まれています。

パブリッククラウドのクラスターがエンドポイントを外部に公開するのを防ぐ必要がある場合、AWS、Azure、または GCP のインストーラーでプロビジョニングされるインフラストラクチャーを使用してプライベートクラスターをデプロイすることができます。

非接続のクラスターまたはネットワークが制限されたクラスターなど、インターネットへのアクセスが限定されたクラスターをインストールする必要がある場合、インストールパッケージをミラーリング し、そこからクラスターをインストールできます。user-provisioned infrastructure によるインストールの詳細な手順を実行して、AWS、GCP、IBM Z® または IBM® LinuxONE、RHEL KVM が搭載された IBM Z® または IBM® LinuxONE、IBM Power®、vSphere、または ベアメタル のネットワークが制限された環境にインストールしてください。ネットワークが制限された環境へのクラスターのインストールは、installer-provisioned infrastructure を使用して、AWS、GCP、IBM Cloud®、Nutanix、RHOSP、および vSphere 用の詳細な手順に従って実行することもできます。

クラスターを AWS GovCloud リージョン、AWS China リージョン、または Azure government リージョン にデプロイする必要がある場合は、インストーラーでプロビジョニングされるインフラストラクチャーのインストール時にこれらのカスタムリージョンを設定できます。

インストール中に FIPS 140-2/140-3 検証 のために NIST に提出された RHEL 暗号化ライブラリーを使用するようにクラスターマシンを設定することもできます。



重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

2.2. インストール後のユーザー向けのクラスターの準備

一部の設定は、クラスターのインストールに必須ではありませんが、ユーザーがクラスターにアクセスする前に設定することが推奨されます。クラスターを設定する Operator を カスタマイズ することでクラスター自体をカスタマイズし、クラスターをアイデンティティープロバイダーなどの他の必要なシステムに統合できます。

実稼働クラスターの場合、以下の統合を設定する必要があります。

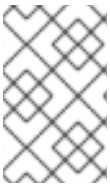
- 永続ストレージ
- アイデンティティープロバイダー
- コア OpenShift Container Platform コンポーネントのモニタリング

2.3. ワークロードについてのクラスターの準備

ワークロードのニーズによっては、アプリケーションのデプロイを開始する前に、追加の手順が必要になる場合があります。たとえば、アプリケーションの ビルドストラテジー をサポートできるようなインフラストラクチャーを準備した後に、低レイテンシー のワークロードに対応できるようにしたり、機密のワークロードを保護 できるようにしたりする必要がある場合があります。アプリケーションワークロードの モニタリング を設定することもできます。

2.4. 各種プラットフォームのサポートされているインストール方法

各種のプラットフォームで各種のインストールを実行できます。



注記

以下の表にあるように、すべてのプラットフォームですべてのインストールオプションがサポートされている訳ではありません。チェックマークは、オプションがサポートされていることを示し、関連するセクションにリンクしています。

表2.1 インストーラーでプロビジョニングされるインフラストラクチャーのオプション

	AWS (64ビット x86)	AWS (64ビット ARM)	Azure (64ビット x86)	Azure (64ビット ARM)	Azure Stack Hub	GCP (64-bit x86)	GCP (64ビット ARM)	Nutanix	RHOSP	ベアメタル (64ビット x86)	ベアメタル (64ビット ARM)	vSphere	IBM Cloud®	IBM Z®	IBM Power®	IBM Power® Virtual Server
デフォルト	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓			
カスタム	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓			✓

	AWS (64ビットx86)	AWS (64ビットARM)	Azure (64ビットx86)	Azure (64ビットARM)	Azure Stack Hub	GCP (64-bit x86)	GCP (64ビットARM)	Nutanix	RHOSP	ベアメタル (64ビットx86)	ベアメタル (64ビットARM)	vSphere	IBM Cloud®	IBM Z®	IBM Power®	IBM Power® Virtual Server
ネットワークのカスタマイズ	✓	✓	✓	✓	✓	✓	✓					✓	✓			
ネットワークが制限されたインストール	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓

	AWS (64ビットx86)	AWS (64ビットARM)	Azure (64ビットx86)	Azure (64ビットARM)	Azure Stack Hub	GCP (64-bit x86)	GCP (64ビットARM)	Nutanix	RHOSP	ベアメタル (64ビットx86)	ベアメタル (64ビットARM)	vSphere	IBM Cloud®	IBM Z®	IBM Power®	IBM Power® Virtual Server
プライベートクラスター	✓	✓	✓	✓		✓	✓						✓			✓
既存の仮想プライベートネットワーク	✓	✓	✓	✓		✓	✓						✓			✓

	AWS (64ビットx86)	AWS (64ビットARM)	Azure (64ビットx86)	Azure (64ビットARM)	Azure Stack Hub	GCP (64-bit x86)	GCP (64ビットARM)	Nutanix	RHOSP	ベアメタル (64ビットx86)	ベアメタル (64ビットARM)	vSphere	IBM Cloud®	IBM Z®	IBM Power®	IBM Power® Virtual Server
government リージョン	✓		✓													
秘密の地域	✓															
China リージョン	✓															

表2.2 ユーザーによってプロビジョニングされるインフラストラクチャーのオプション

	AWS (64ビットx86)	AWS (64ビットARM)	Azure (64ビットx86)	Azure (64ビットARM)	Azure Stack Hub	GCP (64-bit x86)	GCP (64ビットARM)	Nutanix	RHOSP	ベアメタル (64ビットx86)	ベアメタル (64ビットARM)	vSphere	IBM Cloud®	IBM Z®	RHEL KVMを使用したIBM Z®	IBM Power®	プラットフォームの指定なし
カスタム	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	✓	✓
ネットワークのカスタマイズ										✓	✓	✓					
ネットワークが制限されたインストール	✓	✓				✓	✓			✓	✓	✓		✓	✓	✓	

	AWS (64ビットx86)	AWS (64ビットARM)	Azure (64ビットx86)	Azure (64ビットARM)	Azure Stack Hub	GCP (64-bit x86)	GCP (64ビットARM)	Nutanix	RHOSP	ベアメタル (64ビットx86)	ベアメタル (64ビットARM)	vSphere	IBM Cloud®	IBM Z®	RHEL KVMを使用したIBM Z®	IBM Power®	プラットフォームの指定なし
クラスタープロジェクト外でホストされる共有VPC						✓	✓										

第3章 クラスター機能

クラスター管理者は、クラスター機能を使用して、インストール前にオプションコンポーネントを有効または無効にできます。クラスター管理者は、インストール後いつでもクラスター機能を有効にすることができます。

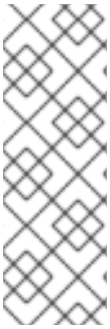
注記

クラスター管理者は、クラスター機能を有効にした後、それを無効にすることはできません。

3.1. クラスター機能の選択

「クラスターをカスタマイズして AWS にインストール」または「クラスターをカスタマイズして GCP にインストール」など、クラスターをカスタマイズしてインストールするいずれかの方法を使用することで、クラスター機能を選択できます。

カスタマイズを伴うインストールでは、クラスターの設定パラメーターを含む **install-config.yaml** ファイルを作成します。



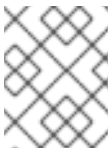
注記

特定のクラスター機能を有効または無効にしてクラスターをカスタマイズする場合は、**install-config.yaml** ファイルを手動で保守する必要があります。新しい OpenShift Container Platform の更新では、既存のコンポーネントの新しい機能ハンドルが宣言されたり、新しいコンポーネントがまとめて導入されたりする可能性があります。**install-config.yaml** ファイルをカスタマイズするユーザーは、OpenShift Container Platform の更新に合わせて **install-config.yaml** ファイルを定期的に更新することも検討してください。

次の設定パラメーターを使用して、クラスター機能を選択できます。

```
capabilities:
  baselineCapabilitySet: v4.11 1
  additionalEnabledCapabilities: 2
  - CSISnapshot
  - Console
  - Storage
```

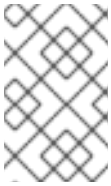
1 インストールする機能のベースラインセットを定義します。有効な値は **None**、**vCurrent**、**v4.x** です。**None** を選択すると、すべてのオプション機能が無効になります。デフォルト値は **vCurrent** で、すべてのオプション機能が有効になります。



注記

v4.x は、現在のクラスターバージョン以前の任意の値を参照します。たとえば、OpenShift Container Platform 4.12 クラスターでの有効値は **v4.11** と **v4.12** です。

2 明示的に有効にする機能のリストを定義します。これらは、**baselineCapabilitySet** で指定された機能に加えて有効になります。



注記

この例では、デフォルトで **v4.11** に設定されています。**additionalEnabledCapabilities** フィールドでは、デフォルトの **v4.11** 機能セット以外を有効にできます。

次の表では、**baselineCapabilitySet** の値について説明します。

表3.1 クラスター機能の baselineCapabilitySet 値の説明

値	説明
vCurrent	新しいリリースで導入される新しいデフォルト機能を自動的に追加する場合、このオプションを指定します。
v4.11	OpenShift Container Platform 4.11 のデフォルト機能を有効にする場合、このオプションを指定します。**v4.11** を指定すると、それ以降のバージョンの OpenShift Container Platform で導入された機能が有効になりません。OpenShift Container Platform 4.11 のデフォルト機能は、**baremetal**、**MachineAPI**、**marketplace**、および **openshift-samples** です。
v4.12	OpenShift Container Platform 4.12 のデフォルト機能を有効にする場合、このオプションを指定します。**v4.12** を指定すると、それ以降のバージョンの OpenShift Container Platform で導入された機能が有効になりません。OpenShift Container Platform 4.12 のデフォルト機能は、**baremetal**、**MachineAPI**、**marketplace**、**openshift-samples**、**Console**、**Insights**、**Storage**、および **CSISnapshot** です。
v4.13	OpenShift Container Platform 4.13 のデフォルト機能を有効にする場合、このオプションを指定します。**v4.13** を指定すると、それ以降のバージョンの OpenShift Container Platform で導入された機能が有効になりません。OpenShift Container Platform 4.13 のデフォルト機能は、**baremetal**、**MachineAPI**、**marketplace**、**openshift-samples**、**Console**、**Insights**、**Storage**、**CSISnapshot**、および **NodeTuning** です。
v4.14	OpenShift Container Platform 4.14 のデフォルト機能を有効にする場合、このオプションを指定します。**v4.14** を指定すると、それ以降のバージョンの OpenShift Container Platform で導入された機能が有効になりません。OpenShift Container Platform 4.14 のデフォルト機能は、**baremetal**、**MachineAPI**、**marketplace**、**openshift-samples**、**Console**、**Insights**、**Storage**、**CSISnapshot**、**NodeTuning**、**ImageRegistry**、**Build**、および **DeploymentConfig** です。

値	説明
v4.15	OpenShift Container Platform 4.15 のデフォルト機能を有効にする場合、このオプションを指定します。**v4.15** を指定すると、それ以降のバージョンの OpenShift Container Platform で導入された機能が有効になりません。OpenShift Container Platform 4.15 のデフォルト機能は、**baremetal**、**MachineAPI**、**marketplace**、**OperatorLifecycleManager**、**openshift-samples**、**Console**、**Insights**、**Storage**、**CSISnapshot**、**NodeTuning**、**ImageRegistry**、**Build**、**CloudCredential**、および **DeploymentConfig** です。
v4.16	OpenShift Container Platform 4.16 のデフォルト機能を有効にする場合、このオプションを指定します。**v4.16** を指定すると、それ以降のバージョンの OpenShift Container Platform で導入された機能が有効になりません。OpenShift Container Platform 4.16 のデフォルトの機能は、**baremetal**、**MachineAPI**、**marketplace**、**OperatorLifecycleManager**、**openshift-samples**、**Console**、**Insights**、**Storage**、**CSISnapshot**、**NodeTuning**、**ImageRegistry**、**Build**、**CloudCredential**、**DeploymentConfig**、および **CloudControllerManager** です。
None	他のセットが大きすぎる場合や、機能が必要ない場合、**additionalEnabledCapabilities** を介して微調整する場合に指定します。

関連情報

- カスタマイズによる AWS へのクラスターのインストール
- カスタマイズによる GCP へのクラスターのインストール

3.2. OPENSHIFT CONTAINER PLATFORM 4.16 のオプションのクラスター機能

現在、クラスター Operator はこれらのオプション機能を提供します。以下は、各オプションによって提供される機能と、無効にした場合に失われる機能をまとめたものです。

関連情報

- クラスター Operator のリファレンス

3.2.1. ベアメタル機能

目的
Cluster Baremetal Operator は、**baremetal** 機能の機能を提供します。

Cluster Baremetal Operator (CBO) は、OpenShift Container Platform コンピュートノードを実行する準備が整った、完全に機能するワーカーノードにベアメタルサーバーを導入するために必要なすべてのコンポーネントをデプロイします。CBO は、Bare Metal Operator (BMO) と Ironic コンテナーで設定される metal3 デプロイメントが、OpenShift Container Platform クラスター内のコントロールプレーンノードの 1 つで実行されるようにします。また、CBO は、監視し、適切なアクションを実行するリソースへの OpenShift Container Platform の更新をリッスンします。

インストーラーによってプロビジョニングされたインフラストラクチャーを使用したデプロイメントには、ベアメタル機能が必要です。ベアメタル機能を無効にすると、これらのデプロイメントで予期しない問題が発生する可能性があります。

クラスター管理者は、クラスター内に **BareMetalHost** リソースを持たないユーザープロビジョニングインフラストラクチャーを使用したインストールの実行中に限り、ベアメタル機能を無効にすることを推奨します。

重要

ベアメタル機能が無効になっている場合、クラスターはベアメタルノードをプロビジョニングまたは管理できません。デプロイメントに **BareMetalHost** リソースがない場合にのみ、この機能を無効にしてください。**baremetal** 機能は **MachineAPI** 機能に依存します。**baremetal** 機能を有効にする場合は、**MachineAPI** も有効にする必要があります。

関連情報

- インストーラーでプロビジョニングされるクラスターのベアメタルへのデプロイ
- ベアメタルクラスターのインストールの準備
- ベアメタルの設定

3.2.2. ビルド機能

目的
Build 機能により、**Build** API が有効になります。**Build** API は、**Build** オブジェクトと **BuildConfig** オブジェクトのライフサイクルを管理します。

重要

Build 機能が無効になっている場合、クラスターは **Build** または **BuildConfig** リソースを使用できません。クラスター内で **Build** リソースおよび **BuildConfig** リソースが必要ない場合にのみ、この機能を無効にします。

3.2.3. Cloud Controller Manager 機能

目的
Cloud Controller Manager Operator は、**CloudControllerManager** の機能を提供します。

注記

CloudControllerManager 機能の無効化は、現在すべてのプラットフォームでサポートされているわけではありません。

クラスターのインストール設定 (**install-config.yaml**) ファイルの値を確認することで、クラスターが **CloudControllerManager** 機能の無効化をサポートしているかどうかを判断できます。

install-config.yaml ファイルで、**platform** パラメーターを見つけます。

- **platform** パラメーターの値が **Baremetal** または **None** の場合、クラスターで **CloudControllerManager** 機能を無効にできます。
- **platform** パラメーターの値が **External** の場合は、**platform.external.cloudControllerManager** パラメーターを見つけます。**platform.external.cloudControllerManager** パラメーターの値が **None** の場合、クラスターで **CloudControllerManager** 機能を無効にできます。

重要

これらのパラメーターに上記以外の値が含まれている場合、クラスターで **CloudControllerManager** 機能を無効にすることはできません。

注記

現在、この Operator は Amazon Web Services (AWS)、Google Cloud Platform (GCP)、IBM Cloud®、グローバル Microsoft Azure、Microsoft Azure Stack Hub、Nutanix、Red Hat OpenStack Platform (RHOSP)、および VMware vSphere 向けに一般提供されています。

Operator は、Alibaba Cloud および IBM Power® Virtual Server で テクノロジープレビュー として利用できます。

Cloud Controller Manager Operator は、OpenShift Container Platform 上にデプロイされた Cloud Controller Manager を管理および更新します。Operator は Kubebuilder フレームワークおよび **controller-runtime** ライブラリーに基づいています。これは Cluster Version Operator (CVO) を使用してインストールされます。

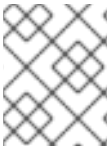
これには、以下のコンポーネントが含まれます。

- Operator
- クラウド設定のオブザーバー

デフォルトで、Operator は **metrics** サービス経由で Prometheus メトリックを公開します。

3.2.4. クラウド認証情報機能

目的
Cloud Credential Operator は、**CloudCredential** の機能を提供します。



注記

現在、**CloudCredential** 機能の無効化は、ベアメタルクラスターでのみサポートされています。

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報を Kubernetes カスタムリソース定義 (CRD) として管理します。CCO は **CredentialsRequest** カスタムリソース (CR) で同期し、OpenShift Container Platform コンポーネントが、クラスターの実行に必要な特定のパーミッションと共にクラウドプロバイダーの認証情報を要求できるようにします。

install-config.yaml ファイルで **credentialsMode** パラメーターに異なる値を設定すると、CCO は複数の異なるモードで動作するように設定できます。モードが指定されていない場合や、**credentialsMode** パラメーターが空の文字列 ("") に設定されている場合は、CCO はデフォルトモードで動作します。

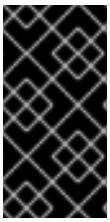
関連情報

- Cloud Credential Operator について

3.2.5. クラスターストレージ機能

目的
Cluster Storage Operator は、**Storage** 機能を提供します。

Cluster Storage Operator は OpenShift Container Platform のクラスター全体のストレージのデフォルト値を設定します。これにより、OpenShift Container Platform クラスターのデフォルトの **storageclass** の存在を確認できます。また、クラスターがさまざまなストレージバックエンドを使用できるようにする Container Storage Interface (CSI) ドライバーもインストールします。



重要

クラスターストレージ機能が無効になっている場合、クラスターにデフォルトの **storageclass** または CSI ドライバーはありません。クラスターストレージ機能が無効になっている場合、管理者権限を持つユーザーは、デフォルトの **storageclass** を作成して CSI ドライバーを手動でインストールできます。

注記

- Operator が作成するストレージクラスは、そのアノテーションを編集することで非デフォルトにすることができますが、Operator が実行されているかぎり、このストレージクラスを削除することはできません。

3.2.6. コンソール機能

目的
Console Operator は、**コンソール** 機能を提供します。

Console Operator は OpenShift Container Platform Web コンソールをクラスターにインストールし、維持します。Console Operator はデフォルトでインストールされ、コンソールを自動的に維持します。

関連情報

- Web コンソールの概要

3.2.7. CSI スナップショットコントローラー機能

目的
Cluster CSI Snapshot Controller Operator は、**CSISnapshot** 機能を提供します。

Cluster CSI Snapshot Controller Operator は、CSI Snapshot Controller をインストールし、維持します。CSI Snapshot Controller は **VolumeSnapshot** CRD オブジェクトを監視し、ボリュームスナップショットの作成および削除のライフサイクルを管理します。

関連情報

- CSI ボリュームスナップショット

3.2.8. DeploymentConfig 機能

目的
DeploymentConfig 機能は、**DeploymentConfig** API を有効にして管理します。

重要

DeploymentConfig 機能を無効にすると、クラスター内で次のリソースが使用できなくなります。

- **DeploymentConfig** リソース
- **deployer** サービスアカウント

クラスターに **DeploymentConfig** リソースと **deployer** サービスアカウントが必要ない場合にのみ、**DeploymentConfig** 機能を無効にしてください。

3.2.9. Ingress Capability

目的
Ingress Operator は、**Ingress** 機能を提供します。

Ingress Operator は OpenShift Container Platform ルーターを設定し、管理します。

プロジェクト
openshift-ingress-operator

CRD

- **clusteringresses.ingress.openshift.io**
  - スコープ: Namespaced
  - CR: **clusteringresses**
  - 検証: No

設定オブジェクト

- クラスター設定
  - タイプ名: **clusteringresses.ingress.openshift.io**
  - インスタンス名: **default**
  - コマンドの表示:

```
$ oc get clusteringresses.ingress.openshift.io -n openshift-ingress-operator default -o yaml
```

注記
Ingress Operator はルーターを **openshift-ingress** プロジェクトに設定し、ルーターのデプロイメントを作成します。

```
$ oc get deployment -n openshift-ingress
```

Ingress Operator は、**network/cluster** ステータスの **clusterNetwork[].cidr** を使用して、管理 Ingress Controller (ルーター) が動作するモード (IPv4、IPv6、またはデュアルスタック) を判別します。たとえば、**clusterNetwork** に v6 **cidr** のみが含まれる場合、Ingress Controller は IPv6 専用モードで動作します。

以下の例では、Ingress Operator によって管理される Ingress Controller は、1 つのクラスターネットワークのみが存在し、ネットワークが IPv4 **cidr** であるために IPv4 専用モードで実行されます。

```
$ oc get network/cluster -o jsonpath='{.status.clusterNetwork[*]}'
```

出力例

```
map[cidr:10.128.0.0/14 hostPrefix:23]
```

3.2.10. Insights 機能

目的
Insights Operator は、**Insights** 機能を提供します。

Insights Operator は OpenShift Container Platform 設定データを収集し、これを Red Hat に送信します。このデータは、クラスターで発生する可能性のある問題について、今後を見据えた上で、事前に対応できる内容に関して推奨事項を生み出します。これらの今後の対応案については、console.redhat.comの Insights Advisor を介してクラスター管理者に伝達されます。

注記
Insights Operator は、OpenShift Container Platform Telemetry を補完します。

関連情報

- Insights Operator の使用

3.2.11. マシン API 機能

目的
machine-api-operator、**cluster-autoscaler-operator**、および **cluster-control-plane-machine-set-operator** Operator は、**MachineAPI** 機能の機能を提供します。この機能は、ユーザーがプロビジョニングしたインフラストラクチャーを使用してクラスターをインストールする場合にのみ無効にできます。

Machine API 機能は、クラスター内のすべてのマシンの設定と管理を担当します。インストール中に Machine API 機能を無効にした場合は、すべてのマシン関連タスクを手動で管理する必要があります。

関連情報

- マシン管理の概要
- Machine API Operator
- Cluster Autoscaler Operator
- Control Plane Machine Set Operator

3.2.12. マーケットプレイス機能

目的
Marketplace Operator は、**marketplace** 機能の機能を提供します。

Marketplace Operator は、クラスター上の一連のデフォルトの Operator Lifecycle Manager (OLM) カタログを使用して、クラスター外の Operator をクラスターに持ち込むプロセスを簡素化します。Marketplace Operator がインストールされると、**openshift-marketplace** namespace が作成されます。OLM は、**openshift-marketplace** namespace にインストールされたカタログソースがクラスター上のすべての namespace で利用可能であることを保証します。

marketplace 機能を無効にすると、Marketplace Operator は **openshift-marketplace** namespace を作成しません。カタログソースは引き続きクラスターで手動で設定および管理できますが、OLM は、クラスター上のすべての namespace でカタログを利用できるようにするために、**openshift-marketplace** namespace に依存しています。システム管理者やクラスター管理者など、**openshift-** で始まる namespace を作成する権限が昇格されたユーザーは、**openshift-marketplace** namespace を手動で作成できます。

marketplace 機能を有効にすると、Marketplace Operator を設定することで個々のカタログを有効または無効にすることができます。

関連情報

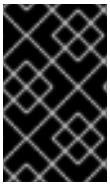
- Red Hat が提供する Operator カタログ

3.2.13. Operator Lifecycle Manager 機能

目的
Operator Lifecycle Manager (OLM) を使用することにより、ユーザーは Kubernetes ネイティブアプリケーション (Operator) および OpenShift Container Platform クラスター全体で実行される関連サービスについてインストール、更新、およびそのライフサイクルの管理を実行できます。これは、Operator を効果的かつ自動化された拡張可能な方法で管理するために設計されたオープンソースツールキットの Operator Framework の一部です。

Operator が次の API のいずれかを必要とする場合は、**OperatorLifecycleManager** 機能を有効にする必要があります。

- **ClusterServiceVersion**
- **CatalogSource**
- **Subscription**
- **InstallPlan**
- **OperatorGroup**



重要

marketplace 機能は、**OperatorLifecycleManager** 機能に依存します。**OperatorLifecycleManager** 機能を無効にして **marketplace** 機能を有効にすることはできません。

関連情報

- Operator Lifecycle Manager の概念およびリソース

3.2.14. ノードチューニング機能

目的
Node Tuning Operator は、**NodeTuning** の機能を提供します。

Node Tuning Operator は、TuneD デーモンを調整することでノードレベルのチューニングを管理し、PerformanceProfile コントローラーを使用して低レイテンシーのパフォーマンスを実現するのに役立ちます。ほとんどの高パフォーマンスアプリケーションでは、一定レベルのカーネルのチューニングが必要です。Node Tuning Operator は、ノードレベルの sysctl の統一された管理インターフェイスをユーザーに提供し、ユーザーが指定するカスタムチューニングを追加できるよう柔軟性を提供します。

NodeTuning 機能を無効にすると、一部のデフォルトチューニング設定がコントロールプレーンノードに適用されなくなります。これにより、900 を超えるノードまたは 900 のルートを持つ大規模なクラスターのスケーラビリティとパフォーマンスが制限される可能性があります。

関連情報

- Node Tuning Operator の使用

3.2.15. OpenShift サンプル機能

目的
Cluster Samples Operator は、**openshift-samples** 機能の機能を提供します。

Cluster Samples Operator は、**openshift** namespace に保存されるサンプルイメージストリームおよびテンプレートを管理します。

初回起動時に、Operator はデフォルトのサンプル設定リソースを作成し、イメージストリームおよびテンプレートの作成を開始します。設定オブジェクトは、キーが **cluster** で、タイプが **configs.samples** のクラススコープのオブジェクトです。

イメージストリームは、**registry.redhat.io** のイメージを参照する Red Hat Enterprise Linux CoreOS (RHCOS) ベースの OpenShift Container Platform イメージストリームです。同様に、テンプレートは OpenShift Container Platform テンプレートとして分類されます。

サンプル機能を無効にすると、ユーザーはそれが提供するイメージストリーム、サンプル、およびテンプレートにアクセスできなくなります。デプロイメントによっては、このコンポーネントが不要な場合は無効にすることができます。

関連情報

- Cluster Samples Operator の設定

3.2.16. クラスターイメージレジストリー機能

目的
Cluster Image Registry Operator は、**ImageRegistry** の機能を提供します。

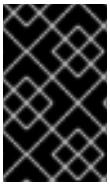
Cluster Image Registry Operator は、OpenShift イメージレジストリーのシングルトンインスタンスを管理します。ストレージの作成を含む、レジストリーのすべての設定を管理します。

初回起動時に、Operator はクラスターで検出される設定に基づいてデフォルトの **image-registry** リソースインスタンスを作成します。これは、クラウドプロバイダーに基づいて使用するクラウドストレージのタイプを示します。

完全な **image-registry** リソースを定義するのに利用できる十分な情報がない場合、その不完全なリソースが定義され、Operator は足りない情報を示す情報を使用してリソースのステータスを更新します。

Cluster Image Registry Operator は **openshift-image-registry** namespace で実行され、その場所のレジストリーインスタンスも管理します。レジストリーのすべての設定およびワークロードリソースはその namespace に置かれます。

イメージレジストリーをクラスターのユーザー認証および認可システムに統合するために、イメージプルシークレットがクラスターのサービスアカウントごとに生成されます。



重要

ImageRegistry 機能を無効にした場合、または Cluster Image Registry Operator の設定で統合済みの OpenShift イメージレジストリーを無効にした場合、イメージプルシークレットはサービスアカウントごとに生成されません。

ImageRegistry 機能を無効にすると、通信環境における OpenShift Container Platform の全体的なリソースフットプリントを削減できます。デプロイメントに応じて、このコンポーネントが必要ない場合は無効にすることができます。

プロジェクト
cluster-image-registry-operator

関連情報

- OpenShift Container Platform のイメージレジストリー Operator
- 自動生成されたシークレット

3.3. 関連情報

- インストール後のクラスター機能の有効化

第4章 非接続インストールミラーリング

4.1. 非接続インストールミラーリングについて

ミラーレジストリーを使用して、クラスターが、外部コンテンツに対する組織の制御条件を満たすコンテナーイメージのみを使用するようにすることができます。ネットワークが制限された環境でプロビジョニングするインフラストラクチャーにクラスターをインストールする前に、必要なコンテナーイメージをその環境にミラーリングする必要があります。コンテナーイメージをミラーリングするには、ミラーリング用のレジストリーが必要です。

4.1.1. ミラーレジストリーの作成

Red Hat Quay などのコンテナーイメージレジストリーがすでにある場合は、それをミラーレジストリーとして使用できます。レジストリーをまだ持っていない場合は、**Red Hat OpenShift のミラーレジストリー** を使用して、ミラーレジストリーを作成 できます。

4.1.2. 非接続インストールのイメージのミラーリング

以下の手順のいずれかを使用して、OpenShift Container Platform イメージリポジトリーをミラーレジストリーにミラーリングできます。

- 非接続インストールのイメージのミラーリング
- oc-mirror プラグインを使用した非接続インストールのイメージのミラーリング

4.2. RED HAT OPENSHIFT 導入用のミラーレジストリーを使用したミラーレジストリーの作成

Red Hat Openshift 導入用のミラーレジストリー は、切断されたインストールに必要な OpenShift Container Platform のコンテナーイメージのミラーリングターゲットとして使用できる小規模で合理化されたコンテナーレジストリーです。

Red Hat Quay などのコンテナーイメージレジストリーがすでにある場合は、このセクションをスキップして、OpenShift Container Platform イメージリポジトリーのミラーリング に直接進むことができます。

4.2.1. 前提条件

- OpenShift Container Platform サブスクリプション
- Podman 3.4.2 以降および OpenSSL がインストールされている Red Hat Enterprise Linux (RHEL) 8 および 9。
- Red Hat Quay サービスの完全修飾ドメイン名。DNS サーバーを介して解決する必要があります。
- ターゲットホストでのキーベースの SSH 接続。SSH キーは、ローカルインストール用に自動的に生成されます。リモートホストの場合は、独自の SSH キーを生成する必要があります。
- vCPU 2 つ以上。
- RAM 8 GB。

- OpenShift Container Platform 4.16 リリースイメージ用に約 12 GB、または OpenShift Container Platform 4.16 リリースイメージおよび OpenShift Container Platform 4.16 Red Hat Operator イメージ用に約 358 GB。ストリームあたり最大 1 TB 以上が推奨されます。

重要

これらの要件は、リリースイメージと Operator イメージのみを使用したローカルテスト結果に基づいています。ストレージ要件は、組織のニーズによって異なります。たとえば、複数の z-stream をミラーリングする場合は、より多くのスペースが必要になることがあります。標準の Red Hat Quay 機能 または適切な API コールアウト を使用して、不要なイメージを削除し、スペースを解放できます。

4.2.2. Red Hat OpenShift 導入用のミラーレジストリー

OpenShift Container Platform の切断されたデプロイメントの場合に、クラスターのインストールを実行するためにコンテナーレジストリーが必要です。このようなクラスターで実稼働レベルのレジストリーサービスを実行するには、別のレジストリーデプロイメントを作成して最初のクラスターをインストールする必要があります。**Red Hat Openshift 導入用のミラーレジストリー**は、このニーズに対応し、すべての OpenShift サブスクリプションに含まれています。これは、OpenShift コンソールの **ダウンロード** ページからダウンロードできます。

Red Hat Openshift 導入用のミラーレジストリーを使用すると、ユーザーは、 **mirror-registry** コマンドラインインターフェイス (CLI) ツールを使用して、Red Hat Quay の小規模バージョンとその必要なコンポーネントをインストールできます。**Red Hat OpenShift 導入用のミラーレジストリー**は、事前設定されたローカルストレージとローカルデータベースを使用して自動的にデプロイされます。また、このレジストリーには、自動生成されたユーザー認証情報とアクセス許可も含まれており、単一の入力セットを使用するだけで開始でき、追加の設定を選択する必要はありません。

Red Hat Openshift のミラーレジストリーは、事前に決定されたネットワーク設定を提供し、成功時にデプロイされたコンポーネントの認証情報とアクセス URL を報告します。完全修飾ドメイン名 (FQDN) サービス、スーパーユーザー名とパスワード、カスタム TLS 証明書などのオプションの設定入力のセットも少しだけ含まれています。これにより、ユーザーはコンテナーレジストリーを利用できるため、制限されたネットワーク環境で OpenShift Container Platform を実行するときに、すべての OpenShift Container Platform リリースコンテンツのオフラインミラーを簡単に作成できます。

インストール環境で別のコンテナーレジストリーがすでに使用可能な場合、**Red Hat Openshift 導入用のミラーレジストリー** の使用はオプションです。

4.2.2.1. Red Hat OpenShift のミラーレジストリーに関する制限

Red Hat OpenShift のミラーレジストリー には次の制限が適用されます。

- **Red Hat Openshift のミラーレジストリー**は高可用性レジストリーではなく、ローカルファイルシステムストレージのみがサポートされます。Red Hat Quay や OpenShift Container Platform の内部イメージレジストリーを置き換えることを目的としたものではありません。

- **Red Hat Openshift のミラーレジストリー**は、Release イメージや Red Hat Operator イメージなど、接続されていない OpenShift Container Platform クラスターのインストールに必要なイメージをホストする場合にのみサポートされます。Red Hat Enterprise Linux (RHEL) マシンのローカルストレージを使用して、RHEL でサポートされるストレージは、**Red Hat Openshift 導入用のミラーレジストリー** でサポートされます。

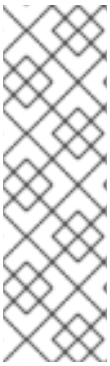
注記

Red Hat OpenShift のミラーレジストリーはローカルストレージを使用します。そのため、イメージをミラーリングするときには、消費されるストレージの使用量に常に注意し、潜在的な問題を軽減するために Red Hat Quay のガベージコレクション機能を使用してください。この機能の詳細は、「Red Hat Quay ガベージコレクション」を参照してください。

- ブートストラップ目的で **Red Hat OpenShift のミラーレジストリー**にプッシュされる Red Hat 製品イメージのサポートは、各製品の有効なサブスクリプションでカバーされます。ブートストラップエクスペリエンスをさらに有効にする例外のリストは、セルフマネージド Red Hat OpenShift のサイジングおよびサブスクリプションガイド に記載されています。
- お客様が作成したコンテンツは、**Red Hat Openshift のミラーレジストリー**でホストしないでください。
- クラスターのグループの更新時にクラスターが複数あると単一障害点を生み出す可能性があるため、複数のクラスターで **Red Hat Openshift のミラーレジストリー**を使用することは推奨されません。**Red Hat Openshift 導入用のミラーレジストリー**を活用して、OpenShift Container Platform コンテンツを他のクラスターに提供できる Red Hat Quay などの実稼働環境レベルの高可用性レジストリーをホストできるクラスターをインストールすることを推奨します。

4.2.3. Red Hat Openshift 導入用のミラーレジストリーを使用したローカルホストでのミラーリング

この手順では、**mirror-registry** インストーラーツールを使用して、 **Red Hat Openshift 導入用のミラーレジストリー**をローカルホストにインストールする方法について説明します。このツールを使用することで、ユーザーは、OpenShift Container Platform イメージのミラーを保存する目的で、ポート 443 で実行されるローカルホストレジストリーを作成できます。



注記

mirror-registry CLI ツールを使用して**Red Hat Openshift 導入用のミラーレジストリー**をインストールすると、マシンにいくつかの変更が加えられます。インストール後、インストールファイル、ローカルストレージ、および設定バンドルを含む **$HOME/quay-install** ディレクトリーが作成されます。デプロイ先がローカルホストである場合には、信頼できる SSH キーが生成され、コンテナーのランタイムが永続的になるようにホストマシン上の systemd ファイルが設定されます。さらに、 **init** という名前の初期ユーザーが、自動生成されたパスワードを使用して作成されます。すべてのアクセス認証情報は、インストール操作の最後に出力されます。

手順

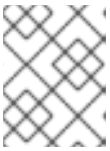
1. OpenShift コンソールの**ダウンロード**ページにある **Red Hat Openshift 導入用のミラーレジストリー**の最新バージョンは、 **mirror-registry.tar.gz** パッケージをダウンロードしてください。
2. **mirror-registry** ツールを使用して、現在のユーザーアカウントでローカルホストに**Red Hat Openshift 導入用のミラーレジストリー** をインストールします。使用可能なフラグの完全なリストは、Red Hat OpenShift フラグのミラーレジストリーを参照してください。

```
$ ./mirror-registry install \
  --quayHostname <host_example_com> \
  --quayRoot <example_directory_name>
```

3. インストール中に生成されたユーザー名とパスワードを使用して、次のコマンドを実行してレジストリーにログインします。

```
$ podman login -u init \
  -p <password> \
  <host_example_com>:8443> \
  --tls-verify=false 1
```

1 生成された root CA 証明書を信頼するようにシステムを設定して、**--tls-verify=false** の実行を回避できます。詳細は、SSL を使用した Red Hat Quay への接続の保護および認証局を信頼するようにシステムを設定するを参照してください。



注記

インストール後、 **https:// <host.example.com>:8443** の UI にアクセスしてログインすることもできます。

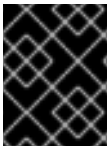
4. ログイン後、OpenShift Container Platform イメージをミラーリングできます。必要に応じて、このドキュメントの OpenShift Container Platform イメージリポジトリーのミラーリングまたは、非接続クラスターで使用する Operator カタログのミラーリングセクションを参照してください。

注記

ストレージレイヤーの問題が原因で**Red Hat Openshift 導入用のミラーレジストリー** で保存されたイメージに問題がある場合は、OpenShift Container Platform イメージを再ミラーリングするか、より安定したストレージにミラーレジストリーを再インストールできます。

4.2.4. ローカルホストからの Red Hat Openshift 導入用のミラーレジストリーの更新

この手順では、**upgrade** コマンドを使用してローカルホストから **Red Hat Openshift 導入用のミラーレジストリー** を更新する方法について説明します。最新バージョンに更新することで、新機能、バグ修正およびセキュリティー脆弱性の修正が保証されます。



重要

更新時には、更新プロセスで再起動されるため、ミラーレジストリーが断続的にダウンします。

前提条件

- **Red Hat Openshift 導入用のミラーレジストリー** をローカルホストにインストールしている。

手順

- **Red Hat OpenShift のミラーレジストリー** を 1.2.z → 1.3.0 にアップグレードし、インストールディレクトリーがデフォルトの **/etc/quay-install** である場合は、次のコマンドを入力できます。

```
$ sudo ./mirror-registry upgrade -v
```

注記

- **Red Hat OpenShift のミラーレジストリー** は、Quay ストレージ、Postgres データ、および **/etc/quay-install** データの Podman ボリュームを新しい **$HOME/quay-install** の場所に移行します。これにより、今後のアップグレード時に **--quayRoot** フラグなしで **Red Hat OpenShift のミラーレジストリー** を使用できます。

- **./mirr-registry upgrade -v** フラグを使用して **Red Hat OpenShift のミラーレジストリー** をアップグレードする場合は、ミラーレジストリーの作成時に使用したものと同じ認証情報を含める必要があります。たとえば、**Red Hat Openshift 導入用のミラーレジストリー** を **--quayHostname<host_example_com>** および **--quayRoot<example_directory_name>** でインストールした場合、ミラーレジストリーを適切にアップグレードするには、その文字列を含める必要があります。

- **Red Hat OpenShift のミラーレジストリー** を 1.2.z から 1.3.0 にアップグレードし、1.2.z デプロイメントで指定されたディレクトリーを使用した場合は、新しい **--pgStorage** フラグと **--quayStorage** フラグを渡す必要があります。以下に例を示します。

```
$ sudo ./mirror-registry upgrade --quayHostname <host_example_com> --quayRoot
<example_directory_name> --pgStorage <example_directory_name>/pg-data --quayStorage
<example_directory_name>/quay-storage -v
```

4.2.5. Red Hat Openshift 導入用のミラーレジストリーを使用したリモートホストでのミラーリング

この手順では、**mirror-registry** ツールを使用して、 **Red Hat Openshift 導入用のミラーレジストリー**をリモートホストにインストールする方法について説明します。そうすることで、ユーザーは OpenShift Container Platform イメージのミラーを保持するレジストリーを作成できます。

注記

mirror-registry CLI ツールを使用して**Red Hat Openshift 導入用のミラーレジストリー**をインストールすると、マシンにいくつかの変更が加えられます。インストール後、インストールファイル、ローカルストレージ、および設定バンドルを含む **$HOME/quay-install** ディレクトリーが作成されます。デプロイ先がローカルホストである場合には、信頼できる SSH キーが生成され、コンテナーのランタイムが永続的になるようにホストマシン上の systemd ファイルが設定されます。さらに、 **init** という名前の初期ユーザーが、自動生成されたパスワードを使用して作成されます。すべてのアクセス認証情報は、インストール操作の最後に出力されます。

手順

1. OpenShift コンソールの**ダウンロード**ページにある **Red Hat Openshift 導入用のミラーレジストリー**の最新バージョンは、 **mirror-registry.tar.gz** パッケージをダウンロードしてください。

2. **mirror-registry** ツールを使用して、現在のユーザーアカウントでローカルホストに**Red Hat Openshift 導入用のミラーレジストリー** をインストールします。使用可能なフラグの完全なリストは、Red Hat OpenShift フラグのミラーレジストリーを参照してください。

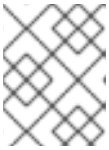
```
$ ./mirror-registry install -v \
  --targetHostname <host_example_com> \
```

```
--targetUsername <example_user> \
-k ~/.ssh/my_ssh_key \
--quayHostname <host_example_com> \
--quayRoot <example_directory_name>
```

3. インストール中に生成されたユーザー名とパスワードを使用して、次のコマンドを実行してミラーレジストリーにログインします。

```
$ podman login -u init \
  -p <password> \
  <host_example_com>:8443> \
  --tls-verify=false 1
```

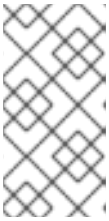
1 生成された root CA 証明書を信頼するようにシステムを設定して、**--tls-verify=false** の実行を回避できます。詳細は、SSL を使用した Red Hat Quay への接続の保護および認証局を信頼するようにシステムを設定するを参照してください。



注記

インストール後、 **https:// <host.example.com>:8443** の UI にアクセスしてログインすることもできます。

4. ログイン後、OpenShift Container Platform イメージをミラーリングできます。必要に応じて、このドキュメントの OpenShift Container Platform イメージリポジトリーのミラーリングまたは、非接続クラスターで使用する Operator カタログのミラーリングセクションを参照してください。

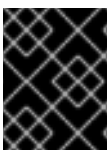


注記

ストレージレイヤーの問題が原因で**Red Hat Openshift 導入用のミラーレジストリー** で保存されたイメージに問題がある場合は、OpenShift Container Platform イメージを再ミラーリングするか、より安定したストレージにミラーレジストリーを再インストールできます。

4.2.6. リモートホストからの Red Hat Openshift 導入用のミラーレジストリーの更新

この手順では、**upgrade** コマンドを使用してリモートホストから **Red Hat Openshift 導入用のミラーレジストリー** を更新する方法について説明します。最新バージョンへの更新により、バグ修正およびセキュリティー脆弱性の修正が確保されます。



重要

更新時には、更新プロセスで再起動されるため、ミラーレジストリーが断続的にダウンします。

前提条件

- **Red Hat Openshift 導入用のミラーレジストリー** をリモートホストにインストールしている。

手順

- **Red Hat Openshift 導入用のミラーレジストリー** をリモートホストからアップグレードするには、以下のコマンドを入力します。

```
$ ./mirror-registry upgrade -v --targetHostname <remote_host_url> --targetUsername
<user_name> -k ~/.ssh/my_ssh_key
```

注記

./mirror-registry upgrade -v フラグを使用して **Red Hat Openshift 導入用のミラーレジストリー** をアップグレードするユーザーは、ミラーレジストリーの作成時に使用したものと同じクレデンシャルを含める必要があります。たとえば、**Red Hat Openshift 導入用のミラーレジストリー** を **--quayHostname<host_example_com>** および **--quayRoot<example_directory_name>** でインストールした場合、ミラーレジストリーを適切にアップグレードするには、その文字列を含める必要があります。

4.2.7. Red Hat OpenShift SSL/TLS 証明書のミラーレジストリーの置き換え

Red Hat OpenShift のミラーレジストリー の SSL/TLS 証明書を更新する必要がある場合もあるはずです。これは、以下のシナリオで役に立ちます。

- 現在の **Red Hat OpenShift のミラーレジストリー** 証明書を置き換える場合。
- 以前の **Red Hat OpenShift のミラーレジストリー** インストールと同じ証明書を使用している場合。
- **Red Hat OpenShift のミラーレジストリー** 証明書を定期的に更新している場合。

Red Hat OpenShift のミラーレジストリー の SSL/TLS 証明書を置き換えるには、次の手順を使用します。

前提条件

- OpenShift コンソールの **ダウンロード** ページから **./mirror-registry** バイナリーをダウンロードしている。

手順

1. 次のコマンドを入力して、**Red Hat OpenShift のミラーレジストリー** をインストールします。

```
$ ./mirror-registry install \
--quayHostname <host_example_com> \
--quayRoot <example_directory_name>
```

Red Hat OpenShift のミラーレジストリー が **$HOME/quay-install** ディレクトリーにインストールされます。

2. 新しい認証局 (CA) バンドルを準備し、新しい **ssl.key** および **ssl.crt** キーファイルを生成します。詳細は、Red Hat Quay への接続を保護するための SSL/TSL の使用 を参照してください。
3. 次のコマンドを入力して、**/$HOME/quay-install** に環境変数 (**QUAY** など) を割り当てます。

```
$ export QUAY=/$HOME/quay-install
```

4. 次のコマンドを入力して、新しい **ssl.crt** ファイルを **/$HOME/quay-install** ディレクトリーにコピーします。

```
$ cp ~/ssl.crt $QUAY/quay-config
```

5. 次のコマンドを入力して、新しい **ssl.key** ファイルを **/$HOME/quay-install** ディレクトリーにコピーします。

```
$ cp ~/ssl.key $QUAY/quay-config
```

6. 次のコマンドを入力して、**quay-app** アプリケーション Pod を再起動します。

```
$ systemctl restart quay-app
```

4.2.8. Red Hat Openshift 導入用のミラーレジストリーのアンインストール

- 次のコマンドを実行して、ローカルホストから **Red Hat Openshift 導入用のミラーレジストリー** をアンインストールできます。

```
$ ./mirror-registry uninstall -v \
  --quayRoot <example_directory_name>
```

注記

- **Red Hat Openshift 導入用のミラーレジストリー** を削除しようとと、削除前にユーザーにプロンプトが表示されます。**--auto Approve** を使用して、このプロンプトをスキップできます。
- **--quayRoot** フラグを指定して **Red Hat Openshift 導入用のミラーレジストリー** をインストールした場合には、アンインストール時に **--quayRoot** フラグを含める必要があります。たとえば、 **Red Hat Openshift 導入用のミラーレジストリー** のインストールで **--quay Rootexample_directory_name** を指定した場合には、 この文字列を追加して、ミラーレジストリーを適切にアンインストールする必要があります。

4.2.9. Red Hat OpenShift フラグのミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー では、以下のフラグを使用できます。

Flags	説明
--autoApprove	対話型プロンプトを無効にするブール値。**true** に設定すると、ミラーレジストリーをアンインストールするときに **quayRoot** ディレクトリーが自動的に削除されます。指定しない場合には、デフォルトは **false** に設定されます。
--initPassword	Quay のインストール中に作成された init ユーザーのパスワード。空白を含まず、8 文字以上にする必要があります。
--initUser string	初期ユーザーのユーザー名を表示します。指定しない場合、デフォルトで **init** になります。

Flags	説明
--no-color, **-c**	インストール、アンインストール、およびアップグレードコマンドの実行時に、ユーザーがカラーシーケンスを無効にして、それを Ansible に伝播できるようにします。
--pgStorage	Postgres 永続ストレージデータの保存先のフォルダー。デフォルトは **pg-storage** Podman ボリュームです。アンインストールには root 権限が必要です。
--quayHostname	クライアントがレジストリーへの接続に使用するミラーレジストリーの完全修飾ドメイン名。**Quayconfig.yaml** の **SERVER_HOSTNAME** に相当します。DNS で解決する必要があります。指定しない場合は、デフォルトは **<target Hostname>:8443** です。[1]
--quayStorage	Quay 永続ストレージデータが保存されるフォルダー。デフォルトは **quay-storage** Podman ボリュームです。アンインストールには root 権限が必要です。
--quayRoot, **-r**	**root CA.key**、**root CA.pem**、**root CA.srl** 証明書など、コンテナーイメージレイヤーと設定データが保存されるディレクトリー。指定しない場合、デフォルトは **$HOME/quay-install** です。
--ssh-key, **-k**	SSH ID キーのパス。指定しない場合、デフォルトは **~/.ssh/quay_installer** です。
--sslCert	SSL/TLS 公開鍵/証明書へのパス。デフォルトは **{quay Root}/quady-config** で、指定しない場合は自動生成されます。
--sslCheckSkip	**config.yaml** ファイルの **SERVER_HOSTNAME** に対する証明書のホスト名のチェックをスキップします。[2]
--sslKey	HTTPS 通信に使用される SSL/TLS 秘密鍵へのパス。デフォルトは **{quay Root}/quady-config** で、指定しない場合は自動生成されます。
--targetHostname, **-H**	Quay のインストール先のホスト名。デフォルトは **$HOST** になります。たとえば、指定していない場合にはローカルホストになります。
--targetUsername, **-u**	SSH に使用するターゲットホストのユーザー。デフォルトは **$USER** です。たとえば、指定しない場合は現在のユーザーになります。
--verbose, **-v**	デバッグログと Ansible Playbook の出力を表示します。
--version	**Red Hat Openshift 導入用のミラーレジストリー** のバージョンを表示します。

1. システムのパブリック DNS 名がローカルホスト名と異なる場合は、**--quayHostname** を変更する必要があります。さらに、**--quayHostname** フラグは、IP アドレスを使用したインストールをサポートしていません。ホスト名を使用してインストールする必要があります。

2. **--ssl Check Skip**は、ミラーレジストリーがプロキシーの背後に設定されており、公開されているホスト名が内部の Quay ホスト名と異なる場合に使用されます。また、インストール中に、指定した Quay ホスト名に対して証明書の検証を行わない場合にも使用できます。

4.2.10. Red Hat OpenShift リリースノートのミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリーは、切断されたインストールに必要な OpenShift Container Platform のコンテナーイメージのミラーリングターゲットとして使用できる小規模で合理化されたコンテナーレジストリーです。

これらのリリースノートは、OpenShift Container Platform で **Red Hat Openshift 導入用のミラーレジストリー** の開発を追跡します。

Red Hat Openshift 導入用のミラーレジストリーの概要については、Red Hat Openshift 導入用のミラーレジストリーを使用したミラーレジストリーの作成 を参照してください。

4.2.10.1. Mirror registry for Red Hat OpenShift 1.3.11

発行日: 2024 年 4 月 23 日

Red Hat OpenShift のミラーレジストリー が、Red Hat Quay 3.8.15 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリーでは、以下のアドバイザリーを使用できます。

- RHBA-2024:1758 - mirror registry for Red Hat OpenShift 1.3.11

4.2.10.2. Mirror registry for Red Hat OpenShift 1.3.10

発行日: 2023 年 12 月 7 日

Red Hat OpenShift のミラーレジストリー が、Red Hat Quay 3.8.14 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリーでは、以下のアドバイザリーを使用できます。

- RHBA-2023:7628 - mirror registry for Red Hat OpenShift 1.3.10

4.2.10.3. Red Hat OpenShift 1.3.9 のミラーレジストリー

発行日: 2023-09-19

Red Hat OpenShift のミラーレジストリー が、Red Hat Quay 3.8.12 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリーでは、以下のアドバイザリーを使用できます。

- RHBA-2023:5241 - mirror registry for Red Hat OpenShift 1.3.9

4.2.10.4. Red Hat OpenShift 1.3.8 のミラーレジストリー

発行日: 2023 年 8 月 16 日

Red Hat OpenShift のミラーレジストリー が、Red Hat Quay 3.8.11 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリーでは、以下のアドバイザリーを使用できます。

- RHBA-2023:4622 - mirror registry for Red Hat OpenShift 1.3.8

4.2.10.5. Red Hat OpenShift 1.3.7 のミラーレジストリー

発行日: 2023-07-19

Red Hat OpenShift のミラーレジストリー が、Red Hat Quay 3.8.10 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2023:4087 - mirror registry for Red Hat OpenShift 1.3.7

4.2.10.6. Red Hat OpenShift 1.3.6 のミラーレジストリー

発行日: 2023-05-30

Red Hat OpenShift のミラーレジストリー が、Red Hat Quay 3.8.8 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2023:3302 - mirror registry for Red Hat OpenShift 1.3.6

4.2.10.7. Red Hat OpenShift 1.3.5 のミラーレジストリー

発行日: 2023-05-18

Red Hat OpenShift のミラーレジストリー が、Red Hat Quay 3.8.7 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2023:3225 - mirror registry for Red Hat OpenShift 1.3.5

4.2.10.8. Red Hat OpenShift 1.3.4 のミラーレジストリー

発行日: 2023-04-25

Red Hat OpenShift のミラーレジストリー が、Red Hat Quay 3.8.6 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2023:1914 - Red Hat OpenShift 1.3.4 のミラーレジストリー

4.2.10.9. Red Hat OpenShift 1.3.3 のミラーレジストリー

発行: 2023-04-05

Red Hat OpenShift のミラーレジストリー が Red Hat Quay 3.8.5 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2023:1528 - Red Hat OpenShift 1.3.3 のミラーレジストリー

4.2.10.10. Red Hat OpenShift 1.3.2 のミラーレジストリー

発行: 2023-03-21

Red Hat OpenShift のミラーレジストリー は、Red Hat Quay 3.8.4 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2023:1376 - Red Hat OpenShift 1.3.2 のミラーレジストリー

4.2.10.11. Red Hat OpenShift 1.3.1 のミラーレジストリー

発行日: 2023-03-7

Red Hat OpenShift のミラーレジストリー は、Red Hat Quay 3.8.3 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2023:1086 - Red Hat OpenShift 1.3.1 のミラーレジストリー

4.2.10.12. Red Hat OpenShift 1.3.0 のミラーレジストリー

発行日: 2023-02-20

Red Hat OpenShift のミラーレジストリー が Red Hat Quay 3.8.1 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2023:0558 - Red Hat OpenShift 1.3.0 のミラーレジストリー

4.2.10.12.1. 新機能

- **Red Hat OpenShift のミラーレジストリー** が Red Hat Enterprise Linux (RHEL) 9 インストールでサポートされるようになりました。
- **Red Hat OpenShift ローカルホストインストールのミラーレジストリー** で IPv6 サポートが利用できるようになりました。
  Red Hat OpenShift リモートホストインストールのミラーレジストリー では、IPv6 は現在サポートされていません。
- 新しい機能フラグ **--quayStorage** が追加されました。このフラグを指定すると、Quay 永続ストレージの場所を手動で設定できます。
- 新しい機能フラグ **--pgStorage** が追加されました。このフラグを指定すると、Postgres 永続ストレージの場所を手動で設定できます。
- 以前は、**Red Hat OpenShift のミラーレジストリー** をインストールするには、root 権限 (**sudo**) が必要でした。今回の更新により、**Red Hat OpenShift のミラーレジストリー** をインストールするために、**sudo** は不要になりました。
  Red Hat OpenShift のミラーレジストリー を **sudo** でインストールすると、インストールファイル、ローカルストレージ、および設定バンドルを含む **/etc/quay-install** ディレクトリーが作成されていました。**sudo** 要件の削除により、インストールファイルと設定バンドルが **$HOME/quay-install** にインストールされるようになりました。Postgres や Quay などのローカルストレージは、Podman によって自動的に作成される名前付きボリュームに格納されるようになりました。

  これらのファイルが保存されているデフォルトのディレクトリーを上書きするには、**Red Hat OpenShift のミラーレジストリー** のコマンドライン引数を使用できます。**Red Hat OpenShift コマンドライン引数のミラーレジストリー** の詳細については、**Red Hat OpenShift フラグのミラーレジストリー** を参照してください。

4.2.10.12.2. バグ修正

- 以前のバージョンでは、**Red Hat OpenShift のミラーレジストリー** をアンインストールしようとすると、次のエラーが返される可能性がありました。**["Error: no container with name or ID \"quay-postgres\" found: no such container"], "stdout": "", "stdout_lines": []***今回の更新により、**Red Hat OpenShift サービスのミラーレジストリー** を停止してアンインストールする順序が変更され、**Red Hat OpenShift のミラーレジストリー** をアンインストールするときにエラーが発生しなくなりました。詳細は、PROJQUAY-4629 を参照してください。

4.2.10.13. Red Hat OpenShift 1.2.9 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.10 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:7369 - mirror registry for Red Hat OpenShift 1.2.9

4.2.10.14. Red Hat OpenShift 1.2.8 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.9 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:7065 - mirror registry for Red Hat OpenShift 1.2.8

4.2.10.15. Red Hat OpenShift 1.2.7 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.8 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:6500 - Red Hat OpenShift 1.2.7 のミラーレジストリー

4.2.10.15.1. バグ修正

- 以前は、**getFQDN()** は完全修飾ドメイン名 (FQDN) ライブラリーに依存してその FQDN を決定し、FQDN ライブラリーは **/etc/hosts** フォルダーを直接読み取ろうとしました。その結果、一部の Red Hat Enterprise Linux CoreOS (RHCOS) インストールで、一般的でない DNS 設定を使用すると、FQDN ライブラリーのインストールが失敗し、インストールが中止されました。今回の更新により、**Red Hat Openshift 導入用のミラーレジストリー** は **hostname** を使用して FQDN を決定します。その結果、FQDN ライブラリーはインストールに失敗しません。(PROJQUAY-4139)

4.2.10.16. Red Hat OpenShift 1.2.6 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.7 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:6278 - Red Hat OpenShift 1.2.6 のミラーレジストリー

4.2.10.16.1. 新機能

新しい機能フラグ **--no-color** (**-c**) が追加されました。この機能フラグにより、インストール、アンインストール、およびアップグレードコマンドの実行時に、ユーザーはカラーシーケンスを無効にして、それを Ansible に伝播することができます。

4.2.10.17. Red Hat OpenShift 1.2.5 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.6 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:6071 - Red Hat OpenShift 1.2.5 のミラーレジストリー

4.2.10.18. Red Hat OpenShift 1.2.4 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.5 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:5884 - Red Hat OpenShift 1.2.4 のミラーレジストリー

4.2.10.19. Red Hat OpenShift 1.2.3 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.4 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:5649 - Red Hat OpenShift 1.2.3 のミラーレジストリー

4.2.10.20. Red Hat OpenShift 1.2.2 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.3 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:5501 - Red Hat OpenShift 1.2.2 のミラーレジストリー

4.2.10.21. Red Hat OpenShift 1.2.1 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.2 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:4986 - Red Hat OpenShift 1.2.1 のミラーレジストリー

4.2.10.22. Red Hat OpenShift 1.2.0 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリー が Red Hat Quay 3.7.1 で利用できるようになりました。

Red Hat Openshift 導入用のミラーレジストリー では、以下のアドバイザリーを使用できます。

- RHBA-2022:4986 - Red Hat OpenShift 1.2.0 のミラーレジストリー

4.2.10.22.1. バグ修正

- 以前は、Quay Pod Operator 内で実行されているすべてのコンポーネントとワーカーのログレベルが **DEBUG** に設定されていました。その結果、不要なスペースを消費する大量のトラフィックログが作成されました。今回の更新では、ログレベルがデフォルトで **WARN** に設定され、トラフィック情報を減らして問題のシナリオに焦点を当てています。(PROJQUAY-3504)

4.2.10.23. Red Hat OpenShift 1.1.0 のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリーでは、以下のアドバイザリーを使用できます。

- RHBA-2022:0956 - Red Hat OpenShift 1.1.0 のミラーレジストリー

4.2.10.23.1. 新機能

- 新しいコマンド **mirror-registry upgrade** が追加されました。このコマンドは、設定やデータに干渉することなく、すべてのコンテナーイメージをアップグレードします。

注記

以前に **quayRoot** がデフォルト以外に設定されていた場合は、それをアップグレードコマンドに渡す必要があります。

4.2.10.23.2. バグ修正

- 以前は、**quayHostname** または **targetHostname** がない場合に、ローカルホスト名がデフォルトになることはありませんでした。今回の更新により、**quayHostname** と **targetHostname** がない場合は、ローカルホスト名がデフォルトになります。(PROJQUAY-3079)

- 以前は、コマンド **./mirror-registry --version** が **unknown flag** エラーを返しました。現在は、**./mirror-registry --version** を実行すると、**Red Hat Openshift 導入用のミラーレジストリー** の現行バージョンが返されます。(PROJQUAY-3086)

- 以前は、たとえば **./mirror-registry install --initUser <user_name> --initPassword <password> --verbose** を実行する場合など、ユーザーはインストール中にパスワードを設定できませんでした。今回の更新により、ユーザーはインストール中にパスワードを設定できるようになりました。(PROJQUAY-3149)

- 以前は、**Red Hat Openshift 導入用のミラーレジストリー**は、Pod が破棄された場合に Pod を再作成しませんでした。現在は、Pod が破棄された場合は Pod が再作成されます。(PROJQUAY-3261)

4.2.11. Red Hat OpenShift のミラーレジストリーのトラブルシューティング

Red Hat OpenShift のミラーレジストリーのトラブルシューティングに、ミラーレジストリーによってインストールされた systemd サービスのログを収集すると役立ちます。次のサービスがインストールされます。

- quay-app.service
- quay-postgres.service
- quay-redis.service

- quay-pod.service

前提条件

- **Red Hat OpenShift のミラーレジストリー** がインストールされている。

手順

- root 権限で **Red Hat OpenShift のミラーレジストリー** をインストールした場合は、次のコマンドを入力して systemd サービスのステータス情報を取得できます。

```
$ sudo systemctl status <service>
```

- 標準ユーザーとして **Red Hat OpenShift のミラーレジストリー** をインストールした場合は、次のコマンドを入力して systemd サービスのステータス情報を取得できます。

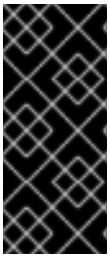
```
$ systemctl --user status <service>
```

4.2.12. 関連情報

- Red Hat Quay ガベージコレクション
- SSL を使用した Red Hat Quay への接続の保護
- 認証局を信頼するようにシステムを設定する
- OpenShift Container Platform イメージリポジトリーのミラーリング
- 非接続クラスターで使用する Operator カタログのミラーリング

4.3. 非接続インストールのイメージのミラーリング

クラスターが、外部コンテンツに対する組織の制限条件を満たすコンテナーイメージのみを使用するようにできますネットワークが制限された環境でプロビジョニングするインフラストラクチャーにクラスターをインストールする前に、必要なコンテナーイメージをその環境にミラーリングする必要があります。コンテナーイメージをミラーリングするには、ミラーリング用のレジストリーが必要です。



重要

必要なコンテナーイメージを取得するには、インターネットへのアクセスが必要です。この手順では、ネットワークとインターネットの両方にアクセスできるミラーホストにミラーレジストリーを配置します。ミラーホストにアクセスできない場合は、非接続クラスターで使用する Operator カタログのミラーリング を使用して、ネットワークの境界を越えて移動できるデバイスにイメージをコピーします。

4.3.1. 前提条件

- 以下のレジストリーのいずれかなど、OpenShift Container Platform クラスターをホストする場所に Docker v2-2 をサポートするコンテナーイメージレジストリーが必要です。
  - Red Hat Quay
  - JFrog Artifactory

- Sonatype Nexus リポジトリー
- Harbor

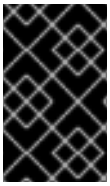
Red Hat Quay のライセンスをお持ちの場合は、概念実証のため に、または Red Hat Quay Operator を使用 して Red Hat Quay をデプロイする方法を記載したドキュメントを参照してください。レジストリーの選択およびインストールがにおいてさらにサポートが必要な場合は、営業担当者または Red Hat サポートにお問い合わせください。

- コンテナーイメージレジストリーの既存のソリューションがまだない場合には、OpenShift Container Platform のサブスクライバーに Red Hat Openshift 導入用のミラーレジストリー が提供されます。**Red Hat Openshift 導入用のミラーレジストリー**はサブスクリプションに含まれており、切断されたインストールで OpenShift Container Platform で必須のコンテナーイメージのミラーリングに使用できる小規模なコンテナーレジストリーです。

4.3.2. ミラーレジストリーについて

OpenShift Container Platform のインストールとその後の製品更新に必要なイメージは、Red Hat Quay、JFrog Artifactory、Sonatype Nexus Repository、Harbor などのコンテナーミラーレジストリーにミラーリングできます。大規模なコンテナーレジストリーにアクセスできない場合は、OpenShift Container Platform サブスクリプションに含まれる小規模なコンテナーレジストリーである **Red Hat OpenShift 導入用のミラーレジストリー** を使用できます。

Red Hat Quay、**Red Hat OpenShift 導入用のミラーレジストリー**、Artifactory、Sonatype Nexus リポジトリー、Harbor など、Dockerv2-2 をサポートする任意のコンテナーレジストリーを使用できます。選択したレジストリーに関係なく、インターネット上の Red Hat がホストするサイトから分離されたイメージレジストリーにコンテンツをミラーリングする手順は同じです。コンテンツをミラーリングした後に、各クラスターをミラーレジストリーからこのコンテンツを取得するように設定します。



重要

OpenShift イメージレジストリーはターゲットレジストリーとして使用できません。これは、ミラーリングプロセスで必要となるタグを使わないプッシュをサポートしないためです。

Red Hat OpenShift 導入用のミラーレジストリー 以外のコンテナーレジストリーを選択する場合は、プロビジョニングするクラスター内の全マシンから到達可能である必要があります。レジストリーに到達できない場合、インストール、更新、またはワークロードの再配置などの通常の操作が失敗する可能性があります。そのため、ミラーレジストリーは可用性の高い方法で実行し、ミラーレジストリーは少なくとも OpenShift Container Platform クラスターの実稼働環境の可用性の条件に一致している必要があります。

ミラーレジストリーを OpenShift Container Platform イメージで設定する場合、2 つのシナリオを実行することができます。インターネットとミラーレジストリーの両方にアクセスできるホストがあり、クラスターノードにアクセスできない場合は、そのマシンからコンテンツを直接ミラーリングできます。このプロセスは、**connected mirroring** (接続ミラーリング) と呼ばれます。このようなホストがない場合は、イメージをファイルシステムにミラーリングしてから、そのホストまたはリムーバブルメディアを制限された環境に配置する必要があります。このプロセスは、**disconnected mirroring** (非接続ミラーリング) と呼ばれます。

ミラーリングされたレジストリーの場合は、プルされたイメージのソースを表示するには、CRI-O ログで **Trying to access** のログエントリーを確認する必要があります。ノードで **crictl images** コマンドを使用するなど、イメージのプルソースを表示する他の方法では、イメージがミラーリングされた場所からプルされている場合でも、ミラーリングされていないイメージ名を表示します。

注記

Red Hat は、OpenShift Container Platform を使用してサードパーティーのレジストリーをテストしません。

関連情報

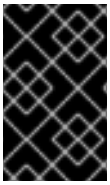
CRI-O ログを表示してイメージソースを表示する方法の詳細は、Viewing the image pull source を参照してください。

4.3.3. ミラーホストの準備

ミラー手順を実行する前に、ホストを準備して、コンテンツを取得し、リモートの場所にプッシュできるようにする必要があります。

4.3.3.1. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.3.4. イメージのミラーリングを可能にする認証情報の設定

Red Hat からミラーにイメージをミラーリングできるコンテナーイメージレジストリー認証情報ファイルを作成します。

警告

クラスターのインストール時に、このイメージレジストリー認証情報ファイルをプルシークレットとして使用しないでください。クラスターのインストール時にこのファイルを指定すると、クラスター内のすべてのマシンにミラーレジストリーへの書き込みアクセスが付与されます。

警告

このプロセスでは、ミラーレジストリーのコンテナーイメージレジストリーへの書き込みアクセスがあり、認証情報をレジストリープルシークレットに追加する必要があります。

前提条件

- 非接続環境で使用するミラーレジストリーを設定しました。
- イメージをミラーリングするミラーレジストリー上のイメージリポジトリーの場所を特定している。
- イメージのイメージリポジトリーへのアップロードを許可するミラーレジストリーアカウントをプロビジョニングしている。

手順

インストールホストで以下の手順を実行します。

1. **registry.redhat.io** プルシークレットを Red Hat OpenShift Cluster Manager からダウンロードします。
2. JSON 形式でプルシークレットのコピーを作成します。

```
$ cat ./pull-secret | jq . > <path>/<pull_secret_file_in_json> 1
```

1 プルシークレットを保存するフォルダーへのパスおよび作成する JSON ファイルの名前を指定します。

ファイルの内容は以下の例のようになります。

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

1. ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードまたはトークンを生成します。

```
$ echo -n '<user_name>:<password>' | base64 -w0 1
BGVtbYk3ZHAtqXs=
```

1 **<user_name>** および **<password>** については、レジストリーに設定したユーザー名およびパスワードを指定します。

2. JSON ファイルを編集し、レジストリーについて記述するセクションをこれに追加します。

```
  "auths": {
    "<mirror_registry>": { 1
      "auth": "<credentials>", 2
      "email": "you@example.com"
    }
  },
```

1 ミラーレジストリーがコンテンツを提供するために使用するレジストリーのドメイン名およびポート (オプション) を指定します。例: **registry.example.com** または **registry.example.com:8443**

2 ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

ファイルは以下の例のようになります。

```
{
  "auths": {
    "registry.example.com": {
      "auth": "BGVtbYk3ZHAtqXs=",
      "email": "you@example.com"
    },
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

4.3.5. OpenShift Container Platform イメージリポジトリーのミラーリング

クラスターのインストールまたはアップグレード時に使用するために、OpenShift Container Platform イメージリポジトリーをお使いのレジストリーにミラーリングします。

前提条件

- ミラーホストがインターネットにアクセスできる。
- ネットワークが制限された環境で使用するミラーレジストリーを設定し、設定した証明書および認証情報にアクセスできる。
- Red Hat OpenShift Cluster Manager からプルシークレット をダウンロードし、ミラーリポジトリーへの認証を組み込むように変更している。
- 自己署名証明書を使用する場合は、証明書にサブジェクトの別名を指定しています。

手順

ミラーホストで以下の手順を実行します。

1. OpenShift Container Platform ダウンロード ページを確認し、インストールする必要のある OpenShift Container Platform のバージョンを判別し、Repository Tags ページで対応するタグを判別します。
2. 必要な環境変数を設定します。
   a. リリースバージョンをエクスポートします。

```
$ OCP_RELEASE=<release_version>
```

<release_version> について、インストールする OpenShift Container Platform のバージョンに対応するタグを指定します (例: **4.5.4**)。

b. ローカルレジストリー名とホストポートをエクスポートします。

```
$ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
```

<local_registry_host_name> については、ミラーレジストリーのレジストリードメイン名を指定し、**<local_registry_host_port>** については、コンテンツの送信に使用するポートを指定します。

c. ローカルリポジトリー名をエクスポートします。

```
$ LOCAL_REPOSITORY='<local_repository_name>'
```

<local_repository_name> については、**ocp4/openshift4** などのレジストリーに作成するリポジトリーの名前を指定します。

d. ミラーリングするリポジトリーの名前をエクスポートします。

```
$ PRODUCT_REPO='openshift-release-dev'
```

実稼働環境のリリースの場合には、**openshift-release-dev** を指定する必要があります。

e. パスをレジストリープルシークレットにエクスポートします。

```
$ LOCAL_SECRET_JSON='<path_to_pull_secret>'
```

<path_to_pull_secret> については、作成したミラーレジストリーのプルシークレットの絶対パスおよびファイル名を指定します。

f. リリースミラーをエクスポートします。

```
$ RELEASE_NAME="ocp-release"
```

実稼働環境のリリースについては、**ocp-release** を指定する必要があります。

g. クラスターのアーキテクチャーのタイプをエクスポートします。

```
$ ARCHITECTURE=<cluster_architecture> 1
```

1 **x86_64**、**aarch64**、**s390x**、または **ppc64le** など、クラスターのアーキテクチャーを指定します。

h. ミラーリングされたイメージをホストするためにディレクトリーへのパスをエクスポートします。

```
$ REMOVABLE_MEDIA_PATH=<path> 1
```

1 最初のスラッシュ (/) 文字を含む完全パスを指定します。

3. バージョンイメージをミラーレジストリーにミラーリングします。

- ミラーホストがインターネットにアクセスできない場合は、以下の操作を実行します。

i. リムーバブルメディアをインターネットに接続しているシステムに接続します。

ii. ミラーリングするイメージおよび設定マニフェストを確認します。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE} --dry-run
```

iii. 直前のコマンドの出力の **imageContentSources** セクション全体を記録します。ミラーの情報はミラーリングされたリポジトリーに一意であり、インストール時に **imageContentSources** セクションを **install-config.yaml** ファイルに追加する必要があります。

iv. イメージをリムーバブルメディア上のディレクトリーにミラーリングします。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}
```

v. メディアをネットワークが制限された環境に移し、イメージをローカルコンテナーレジストリーにアップロードします。

```
$ oc image mirror -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} (1)
```

(1) **REMOVABLE_MEDIA_PATH** の場合、イメージのミラーリング時に指定した同じパスを使用する必要があります。

重要

oc image mirror を実行すると、**error: unable to retrieve source image** エラーが発生する場合があります。このエラーは、イメージレジストリーに存在しなくなったイメージへの参照がイメージインデックスに含まれている場合に発生します。イメージインデックスは、それらのイメージを実行しているユーザーがアップグレードグラフの新しいポイントへのアップグレードパスを実行できるように、古い参照を保持する場合があります。一時的な回避策として、**--skip-missing** オプションを使用してエラーを回避し、イメージインデックスのダウンロードを続行できます。詳細は、Service Mesh Operator mirroring failed を参照してください。

- ローカルコンテナーレジストリーがミラーホストに接続されている場合は、以下の操作を実行します。

i. 以下のコマンドを使用して、リリースイメージをローカルレジストリーに直接プッシュします。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
```

このコマンドは、リリース情報をダイジェストとしてプルします。その出力には、クラスターのインストール時に必要な **imageContentSources** データが含まれます。

ii. 直前のコマンドの出力の **imageContentSources** セクション全体を記録します。ミラーの情報はミラーリングされたリポジトリーに一意であり、インストール時に **imageContentSources** セクションを **install-config.yaml** ファイルに追加する必要があります。

注記

ミラーリングプロセス中にイメージ名に Quay.io のパッチが適用され、podman イメージにはブートストラップ仮想マシンのレジストリーに Quay.io が表示されます。

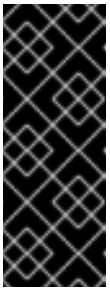
4. ミラーリングしたコンテンツに基づくインストールプログラムを作成するために、インストールプログラムを展開してリリースに固定します。

- ミラーホストがインターネットにアクセスできない場合は、以下のコマンドを実行します。

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --icsp-file=<file> \ --
command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}"
```

- ローカルコンテナーレジストリーがミラーホストに接続されている場合は、以下のコマンドを実行します。

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}"
```



重要

選択した OpenShift Container Platform のバージョンに適したイメージを確実に使用するために、ミラーリングしたコンテンツからインストールプログラムを展開する必要があります。

インターネット接続のあるマシンで、このステップを実行する必要があります。

5. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの場合は、以下のコマンドを実行します。

```
$ openshift-install
```

4.3.6. 非接続環境の Cluster Samples Operator

非接続環境で Cluster Samples Operator を設定するには、クラスターのインストール後に追加の手順を実行する必要があります。以下の情報を確認し、準備してください。

4.3.6.1. ミラーリングの Cluster Samples Operator のサポート

インストール時に、OpenShift Container Platform は **imagestreamtag-to-image** という名前の設定マップを **openshift-cluster-samples-operator** namespace に作成します。**imagestreamtag-to-image** 設定マップには、各イメージストリームタグのエントリー (設定されるイメージ) が含まれます。

設定マップの data フィールドの各エントリーのキーの形式は、**<image_stream_name>_<image_stream_tag_name>** です。

OpenShift Container Platform の非接続インストール時に、Cluster Samples Operator のステータスは **Removed** に設定されます。これを **Managed** に変更することを選択する場合、サンプルがインストールされます。

注記

ネットワークが制限されている環境または切断されている環境でサンプルを使用するには、ネットワークの外部のサービスにアクセスする必要がある場合があります。サービスの例には、Github、Maven Central、npm、RubyGems、PyPi などがあります。場合によっては、Cluster Samples Operator のオブジェクトが必要なサービスに到達できるようにするために、追加の手順を実行する必要があります。

この config map は、イメージストリームをインポートするためにミラーリングする必要があるイメージの参照情報として使用できます。

- Cluster Samples Operator が **Removed** に設定される場合、ミラーリングされたレジストリーを作成するか、使用する必要のある既存のミラーリングされたレジストリーを判別できます。
- 新しい config map をガイドとして使用し、ミラーリングされたレジストリーに必要なサンプルをミラーリングします。
- Cluster Samples Operator 設定オブジェクトの **skippedImagestreams** リストに、ミラーリングされていないイメージストリームを追加します。
- Cluster Samples Operator 設定オブジェクトの **samplesRegistry** をミラーリングされたレジストリーに設定します。
- 次に、Cluster Samples Operator を **Managed** に設定し、ミラーリングしたイメージストリームをインストールします。

4.3.7. 非接続クラスターで使用する Operator カタログのミラーリング

oc adm catalog mirror コマンドを使用して、Red Hat が提供するカタログまたはカスタムカタログの Operator コンテンツをコンテナーイメージレジストリーにミラーリングできます。ターゲットレジストリーは Docker v2-2 をサポートする必要があります。ネットワークが制限された環境のクラスターの場合、このレジストリーには、ネットワークが制限されたクラスターのインストール時に作成されたミラーレジストリーなど、クラスターにネットワークアクセスのあるレジストリーを使用できます。

重要

- OpenShift イメージレジストリーはターゲットレジストリーとして使用できません。これは、ミラーリングプロセスで必要となるタグを使わないプッシュをサポートしないためです。
- **oc adm catalog mirror** を実行すると、**error: unable to retrieve source image** エラーが発生する場合があります。このエラーは、イメージレジストリーに存在しなくなったイメージへの参照がイメージインデックスに含まれている場合に発生します。イメージインデックスは、それらのイメージを実行しているユーザーがアップグレードグラフの新しいポイントへのアップグレードパスを実行できるように、古い参照を保持する場合があります。一時的な回避策として、**--skip-missing** オプションを使用してエラーを回避し、イメージインデックスのダウンロードを続行できます。詳細は、Service Mesh Operator mirroring failed を参照してください。

oc adm catalog mirror コマンドは、Red Hat が提供するインデックスイメージであるか、独自のカスタムビルドされたインデックスイメージであるかに関係なく、ミラーリングプロセス中に指定されるインデックスイメージをターゲットレジストリーに自動的にミラーリングします。次に、ミラーリングされたインデックスイメージを使用して、Operator Lifecycle Manager (OLM) がミラーリングされたカタログを OpenShift Container Platform クラスターにロードできるようにするカタログソースを作成できます。

関連情報

- ネットワークが制限された環境での Operator Lifecycle Manager の使用

4.3.7.1. 前提条件

非接続クラスターで使用する Operator カタログのミラーリングには、以下の前提条件があります。

- ネットワークアクセスが無制限のワークステーション
- **podman** バージョン 1.9.3 以降。
- 既存のカタログをフィルタリングまたは **プルーニング** して、Operator のサブセットのみを選択的にミラーリングする場合は、次のセクションを参照してください。
  - opm CLI のインストール
  - ファイルベースのカタログイメージの更新またはフィルタリング
- Red Hat が提供するカタログをミラーリングする場合は、ネットワークアクセスが無制限のワークステーションで以下のコマンドを実行し、**registry.redhat.io** で認証します。

```
$ podman login registry.redhat.io
```

- Docker v2-2 をサポートするミラーレジストリーへのアクセス。
- ミラーレジストリーで、ミラーリングされた Operator コンテンツの保存に使用するリポジトリーまたは namespace を決定します。たとえば、**olm-mirror** リポジトリーを作成できます。
- ミラーレジストリーにインターネットアクセスがない場合は、ネットワークアクセスが無制限のワークステーションにリムーバブルメディアを接続します。

- **registry.redhat.io** などのプライベートレジストリーを使用している場合、後続の手順で使用するために **REG_CREDS** 環境変数をレジストリー認証情報のファイルパスに設定します。たとえば **podman** CLI の場合は、以下のようになります。

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

4.3.7.2. カタログコンテンツの抽出およびミラーリング

oc adm catalog mirror コマンドは、インデックスイメージのコンテンツを抽出し、ミラーリングに必要なマニフェストを生成します。コマンドのデフォルト動作で、マニフェストを生成し、インデックスイメージからのすべてのイメージコンテンツを、インデックスイメージと同様にミラーレジストリーに対して自動的にミラーリングします。

または、ミラーレジストリーが完全に非接続または **エアギャップ**環境のホスト上にある場合、最初にコンテンツをリムーバブルメディアにミラーリングし、メディアを非接続環境に移行してから、メディアからレジストリーにコンテンツをレジストリーに対してミラーリングできます。

4.3.7.2.1. 同じネットワーク上のレジストリーへのカタログコンテンツのミラーリング

ミラーレジストリーがネットワークアクセスが無制限のワークステーションと同じネットワーク上に置かれている場合は、ワークステーションで以下のアクションを実行します。

手順

1. ミラーレジストリーに認証が必要な場合は、以下のコマンドを実行してレジストリーにログインします。

```
$ podman login <mirror_registry>
```

2. 以下のコマンドを実行して、コンテンツをミラーレジストリーに対して抽出し、ミラーリングします。

```
$ oc adm catalog mirror \
    <index_image> \ 1
    <mirror_registry>:<port>[/<repository>] \ 2
    [-a ${REG_CREDS}] \ 3
    [--insecure] \ 4
    [--index-filter-by-os='<platform>/<arch>'] \ 5
    [--manifests-only] 6
```

1. ミラーリングするカタログのインデックスイメージを指定します。
2. Operator の内容をミラーリングするターゲットレジストリーの完全修飾ドメイン名 (FQDN) を指定します。ミラーレジストリー **<repository>** には、前提条件で説明した **olm-mirror** など、レジストリー上の既存のリポジトリーまたは namespace を指定できます。ミラーリング中に既存のリポジトリーが見つかった場合は、そのリポジトリー名が結果のイメージ名に追加されます。イメージ名にリポジトリー名を含めたくない場合は、この行から **<repository>** 値を省略します (例: **<mirror_registry>:<port>**)。
3. オプション: 必要な場合は、レジストリー認証情報ファイルの場所を指定します。**registry.redhat.io** には、**{REG_CREDS}** が必要です。
4. オプション: ターゲットレジストリーの信頼を設定しない場合は、**--insecure** フラグを追加します。

5 オプション: 複数のバリアントが利用可能な場合に、選択するインデックスイメージのプラットフォームおよびアーキテクチャーを指定します。イメージは **'<platform>/<arch>**

6 オプション: 実際にイメージコンテンツをレジストリーにミラーリングせずに、ミラーリングに必要なマニフェストのみを生成します。このオプションは、ミラーリングする内容を確認するのに役立ちます。また、パッケージのサブセットのみが必要な場合に、マッピングのリストに変更を加えることができます。次に、**mapping.txt** ファイルを **oc image mirror** コマンドで使用し、後のステップでイメージの変更済みの一覧をミラーリングできます。これは、カタログからのコンテンツの高度な選択可能ミラーリングの実行に使用するためのフラグです。

出力例

```
src image has index label for database path: /database/index.db
using database path mapping: /database/index.db:/tmp/153048078
wrote database to /tmp/153048078 1
...
wrote mirroring manifests to manifests-redhat-operator-index-1614211642 2
```

1 コマンドで生成された一時的な **index.db** データベースのディレクトリー。

2 生成される manifests ディレクトリー名を記録します。このディレクトリーは、後続の手順で参照されます。

注記

Red Hat Quay では、ネストされたリポジトリーはサポート対象外です。その結果、**oc adm catalog mirror** コマンドを実行すると、**401** unauthorized エラーで失敗します。回避策として、**oc adm catalog mirror** コマンドを実行するときに **--max-components = 2** オプションを使用して、ネストされたリポジトリーの作成を無効にすることができます。この回避策の詳細は Unauthorized error thrown while using catalog mirror command with Quay registry のナレッジソリューションを参照してください。

関連情報

- Operator のアーキテクチャーおよびオペレーティングシステムのサポート

4.3.7.2.2. カタログコンテンツをエアギャップされたレジストリーへのミラーリング

ミラーレジストリーが完全に切断された、またはエアギャップのあるホスト上にある場合は、次のアクションを実行します。

手順

1. ネットワークアクセスが無制限のワークステーションで以下のコマンドを実行し、コンテンツをローカルファイルにミラーリングします。

```
$ oc adm catalog mirror \
    <index_image> \ 1
    file:///local/index \ 2
```

```
-a ${REG_CREDS} \ 3
--insecure \ 4
--index-filter-by-os='<platform>/<arch>' 5
```

1. ミラーリングするカタログのインデックスイメージを指定します。
2. 現在のディレクトリーのローカルファイルにミラーリングするコンテンツを指定します。
3. オプション: 必要な場合は、レジストリー認証情報ファイルの場所を指定します。
4. オプション: ターゲットレジストリーの信頼を設定しない場合は、**--insecure** フラグを追加します。
5. オプション: 複数のバリアントが利用可能な場合に、選択するインデックスイメージのプラットフォームおよびアーキテクチャーを指定します。イメージは **'<platform>/<arch>[/<variant>]'** として指定されます。これはインデックスで参照されるイメージには適用されません。使用できる値は、**linux/amd64**、**linux/ppc64le**、**linux/s390x**、**linux/arm64**、および **.***です。

出力例

```
...
info: Mirroring completed in 5.93s (5.915MB/s)
wrote mirroring manifests to manifests-my-index-1614985528 1

To upload local images to a registry, run:

 oc adm catalog mirror file://local/index/myrepo/my-index:v1 REGISTRY/REPOSITORY 2
```

1. 生成される manifests ディレクトリー名を記録します。このディレクトリーは、後続の手順で参照されます。
2. 提供されたインデックスイメージをベースとする、拡張された **file://** パスを記録します。このパスは、後続のステップで参照されます。

このコマンドにより、現在のディレクトリーに **v2/** ディレクトリーが作成されます。

2. **v2/** ディレクトリーをリムーバブルメディアにコピーします。
3. メディアを物理的に削除して、これをミラーレジストリーにアクセスできる非接続環境のホストに割り当てます。
4. ミラーレジストリーに認証が必要な場合は、非接続環境のホストで以下のコマンドを実行し、レジストリーにログインします。

```
$ podman login <mirror_registry>
```

5. **v2/** ディレクトリーを含む親ディレクトリーから以下のコマンドを実行し、ローカルファイルからミラーレジストリーにイメージをアップロードします。

```
$ oc adm catalog mirror \
    file://local/index/<repository>/<index_image>:<tag> \ 1
    <mirror_registry>:<port>[/<repository>] \ 2
```

```
-a ${REG_CREDS} \ 3
--insecure \ 4
--index-filter-by-os='<platform>/<arch>' 5
```

1. 直前のコマンド出力の **file://** パスを指定します。
2. Operator の内容をミラーリングするターゲットレジストリーの完全修飾ドメイン名 (FQDN) を指定します。ミラーレジストリー **<repository>** には、前提条件で説明した **olm-mirror** など、レジストリー上の既存のリポジトリーまたは namespace を指定できます。ミラーリング中に既存のリポジトリーが見つかった場合は、そのリポジトリー名が結果のイメージ名に追加されます。イメージ名にリポジトリー名を含めたくない場合は、この行から **<repository>** 値を省略します (例: **<mirror_registry>:<port>)**。
3. オプション: 必要な場合は、レジストリー認証情報ファイルの場所を指定します。
4. オプション: ターゲットレジストリーの信頼を設定しない場合は、**--insecure** フラグを追加します。
5. オプション: 複数のバリアントが利用可能な場合に、選択するインデックスイメージのプラットフォームおよびアーキテクチャーを指定します。イメージは **'<platform>/<arch>[/<variant>]'** として指定されます。これはインデックスで参照されるイメージには適用されません。使用できる値は、**linux/amd64**、**linux/ppc64le**、**linux/s390x**、**linux/arm64**、および **.***です。

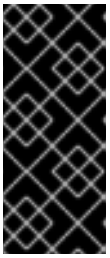
注記

Red Hat Quay では、ネストされたリポジトリーはサポート対象外です。その結果、**oc adm catalog mirror** コマンドを実行すると、**401** unauthorized エラーで失敗します。回避策として、**oc adm catalog mirror** コマンドを実行するときに **--max-components = 2** オプションを使用して、ネストされたリポジトリーの作成を無効にすることができます。この回避策の詳細は Unauthorized error thrown while using catalog mirror command with Quay registry のナレッジソリューションを参照してください。

6. **oc adm catalogmirror** コマンドを再度実行します。新しくミラー化されたインデックスイメージをソースとして使用し、前の手順で使用したものと同じミラーレジストリーターゲットを使用します。

```
$ oc adm catalog mirror \
  <mirror_registry>:<port>/<index_image> \
  <mirror_registry>:<port>[/<repository>] \
  --manifests-only \ 1
  [-a ${REG_CREDS}] \
  [--insecure]
```

1. コマンドがミラーリングされたすべてのコンテンツを再度コピーしないように、このステップには **--manifests-only** フラグが必要です。



重要

前のステップで生成された **imageContentSourcePolicy.yaml** ファイルのイメージマッピングをローカルパスから有効なミラー位置に更新する必要があるため、このステップが必要です。そうしないと、後のステップで **imageContentSourcePolicy** オブジェクトを作成するときにエラーが発生します。

カタログのミラーリング後、残りのクラスターインストールを続行できます。クラスターのインストールが正常に完了した後に、この手順から manifests ディレクトリーを指定して **ImageContentSourcePolicy** および **CatalogSource** オブジェクトを作成する必要があります。これらのオブジェクトは、OperatorHub からの Operator のインストールを有効にするために必要になります。

関連情報

- Operator のアーキテクチャーおよびオペレーティングシステムのサポート

4.3.7.3. 生成されたマニフェスト

Operator カタログコンテンツをミラーレジストリーにミラーリングした後に、現在のディレクトリーに manifests ディレクトリーが生成されます。

同じネットワークのレジストリーにコンテンツをミラーリングする場合、ディレクトリー名は以下のパターンになります。

```
manifests-<index_image_name>-<random_number>
```

直前のセクションで非接続ホストのレジストリーにコンテンツをミラーリングする場合、ディレクトリー名は以下のパターンになります。

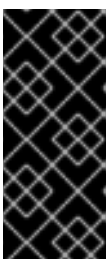
```
manifests-index/<repository>/<index_image_name>-<random_number>
```

注記

manifests ディレクトリー名は、後続の手順で参照されます。

manifests ディレクトリーには以下のファイルが含まれており、これらの一部にはさらに変更が必要になる場合があります。

- **catalogSource.yaml** ファイルは、インデックスイメージタグおよび他の関連するメタデータで事前に設定される **CatalogSource** オブジェクトの基本的な定義です。このファイルは、カタログソースをクラスターに追加するためにそのまま使用したり、変更したりできます。



重要

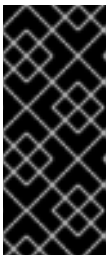
ローカルファイルにコンテンツをミラーリングする場合は、**catalogSource.yaml** ファイルを変更して **metadata.name** フィールドからバックスラッシュ (/) 文字を削除する必要があります。または、オブジェクトの作成を試みると、invalid resource name (無効なリソース名) を示すエラーを出して失敗します。

- これにより、**imageContentSourcePolicy.yaml** ファイルは **ImageContentSourcePolicy** オブジェクトを定義します。このオブジェクトは、ノードを Operator マニフェストおよびミラーリングされたレジストリーに保存されるイメージ参照間で変換できるように設定します。

注記

クラスターが **ImageContentSourcePolicy** オブジェクトを使用してリポジトリーのミラーリングを設定する場合、ミラーリングされたレジストリーにグローバルプルシークレットのみを使用できます。プロジェクトにプルシークレットを追加することはできません。

- **mapping.txt** ファイルには、すべてのソースイメージが含まれ、これはそれらのイメージをターゲットレジストリー内のどこにマップするかを示します。このファイルは **oc image mirror** コマンドと互換性があり、ミラーリング設定をさらにカスタマイズするために使用できます。



重要

ミラーリングのプロセスで **--manifests-only** フラグを使用しており、ミラーリングするパッケージのサブセットをさらにトリミングするには、**mapping.txt** ファイルの変更および **oc image mirror** コマンドでのファイルの使用について、OpenShift Container Platform 4.7 ドキュメントの Package Manifest Format カタログイメージのミラーリング の手順を参照してください。

4.3.7.4. インストール後の要件

カタログのミラーリング後、残りのクラスターインストールを続行できます。クラスターのインストールが正常に完了した後に、この手順から manifests ディレクトリーを指定して **ImageContentSourcePolicy** および **CatalogSource** オブジェクトを作成する必要があります。これらのオブジェクトは、OperatorHub からの Operator のインストールを設定し、有効にするために必要です。

関連情報

- ミラーリングされた Operator カタログからの OperatorHub の入力
- ファイルベースのカタログイメージの更新またはフィルタリング

4.3.8. 次のステップ

- VMware vSphere、ベアメタル、または Amazon Web Services など、ネットワークが制限された環境でプロビジョニングするインフラストラクチャーにクラスターをインストールします。

4.3.9. 関連情報

- must-gather の使用についての詳細は、特定機能に関するデータの収集 を参照してください。

4.4. OC-MIRROR プラグインを使用した非接続インストールのイメージのミラーリング

プライベートレジストリー内の OpenShift Container Platform コンテナーイメージのミラーリングされたセットからクラスターをインストールすることにより、インターネットに直接接続せずに制限されたネットワークでクラスターを実行することができます。このレジストリーは、クラスターが実行されて

いる限り、常に実行されている必要があります。詳細は、前提条件 セクションを参照してください。

oc-mirror OpenShift CLI (**oc**) プラグインを使用して、完全なまたは部分的な非接続環境でイメージをミラーレジストリーにミラーリングできます。公式の Red Hat レジストリーから必要なイメージをダウンロードするには、インターネット接続のあるシステムから oc-mirror を実行する必要があります。

4.4.1. oc-mirror プラグインについて

oc-mirror OpenShift CLI(**oc**) プラグインを使用すると、単一のツールを使用して、必要なすべての OpenShift Container Platform コンテンツおよびその他のイメージをミラーレジストリーにミラーリングできます。次の機能を提供します。

- OpenShift Container Platform のリリース、Operator、ヘルムチャート、およびその他のイメージをミラーリングするための一元化された方法を提供します。
- OpenShift Container Platform および Operator の更新パスを維持します。
- 宣言型イメージセット設定ファイルを使用して、クラスターに必要な OpenShift Container Platform リリース、Operator、およびイメージのみを含めます。
- 将来のイメージセットのサイズを縮小するインクリメンタルミラーリングを実行します。
- 前回の実行以降にイメージセット設定から除外されたターゲットミラーレジストリーからのイメージをプルーニングします。
- オプションで、OpenShift Update Service (OSUS) を使用する際のサポートアーティファクトを生成します。

oc-mirror プラグインを使用する場合、イメージセット設定ファイルでミラーリングするコンテンツを指定します。この YAML ファイルでは、クラスターに必要な OpenShift Container Platform リリースと Operator のみを含めるように設定を微調整できます。これにより、ダウンロードして転送する必要のあるデータの量が減ります。oc-mirror プラグインは、任意のヘルムチャートと追加のコンテナーイメージをミラーリングして、ユーザーがワークロードをミラーレジストリーにシームレスに同期できるようにすることもできます。

oc-mirror プラグインを初めて実行すると、非接続クラスターのインストールまたは更新を実行するために必要なコンテンツがミラーレジストリーに入力されます。非接続クラスターが更新を受信し続けるには、ミラーレジストリーを更新しておく必要があります。ミラーレジストリーを更新するには、最初に実行したときと同じ設定を使用して oc-mirror プラグインを実行します。oc-mirror プラグインは、ストレージバックエンドからメタデータを参照し、ツールを最後に実行してからリリースされたもののみをダウンロードします。これにより、OpenShift Container Platform および Operator の更新パスが提供され、必要に応じて依存関係の解決が実行されます。

4.4.1.1. ワークフローの概要

次の手順は、oc-mirror プラグインを使用してイメージをミラーレジストリーにミラーリングする方法の概要を示しています。

1. イメージセット設定ファイルを作成します。
2. 次のいずれかの方法を使用して、イメージセットをターゲットミラーレジストリーにミラーリングします。
    - イメージセットをターゲットミラーレジストリーに直接ミラーリングします。
    - イメージセットをディスクにミラーリングし、イメージセットをターゲット環境に転送してから、イメージセットをターゲットミラーレジストリーにアップロードします。

3. oc-mirror プラグインが生成したリソースを使用するようにクラスターを設定します。

4. 必要に応じてこれらの手順を繰り返して、ターゲットミラーレジストリーを更新します。

重要

oc-mirror CLI プラグインを使用してミラーレジストリーに入力した場合、ターゲットミラーレジストリーへのそれ以降の更新は、oc-mirror プラグインを使用して行う必要があります。

4.4.2. oc-mirror プラグインの互換性とサポート

oc-mirror プラグインは、OpenShift Container Platform バージョン 4.10 以降の OpenShift Container Platform ペイロードイメージと Operator カタログのミラーリングをサポートします。

注記

aarch64、**ppc64le**、および **s390x** アーキテクチャーでは、oc-mirror プラグインは OpenShift Container Platform バージョン 4.14 以降でのみサポートされます。

ミラーリングする必要がある OpenShift Container Platform のバージョンに関係なく、使用可能な最新バージョンの oc-mirror プラグインを使用してください。

関連情報

- oc-mirror の更新については、イメージのプルソースの表示 を参照してください。

4.4.3. ミラーレジストリーについて

OpenShift Container Platform のインストールとその後の製品更新に必要なイメージを、Red Hat Quay などの Docker v2-2 をサポートするコンテナーミラーレジストリーにミラーリングできます。大規模なコンテナーレジストリーにアクセスできない場合は、OpenShift Container Platform サブスクリプションに含まれる小規模なコンテナーレジストリーである**Red Hat Openshift 導入用のミラーレジストリー**を使用できます。

選択したレジストリーに関係なく、インターネット上の Red Hat がホストするサイトから分離されたイメージレジストリーにコンテンツをミラーリングする手順は同じです。コンテンツをミラーリングした後に、各クラスターをミラーレジストリーからこのコンテンツを取得するように設定します。

重要

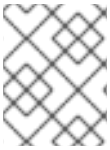
OpenShift イメージレジストリーはターゲットレジストリーとして使用できません。これは、ミラーリングプロセスで必要となるタグを使わないプッシュをサポートしないためです。

Red Hat OpenShift 導入用のミラーレジストリー以外のコンテナーレジストリーを選択する場合は、プロビジョニングするクラスター内の全マシンから到達可能である必要があります。レジストリーに到達できない場合、インストール、更新、またはワークロードの再配置などの通常の操作が失敗する可能性があります。そのため、ミラーレジストリーは可用性の高い方法で実行し、ミラーレジストリーは少なくとも OpenShift Container Platform クラスターの実稼働環境の可用性の条件に一致している必要があります。

ミラーレジストリーを OpenShift Container Platform イメージで設定する場合、2 つのシナリオを実行

することができます。インターネットとミラーレジストリーの両方にアクセスできるホストがあり、クラスターノードにアクセスできない場合は、そのマシンからコンテンツを直接ミラーリングできます。このプロセスは、**connected mirroring** (接続ミラーリング) と呼ばれます。このようなホストがない場合は、イメージをファイルシステムにミラーリングしてから、そのホストまたはリムーバブルメディアを制限された環境に配置する必要があります。このプロセスは、**disconnected mirroring** (非接続ミラーリング) と呼ばれます。

ミラーリングされたレジストリーの場合は、プルされたイメージのソースを表示するには、CRI-O ログで **Trying to access** のログエントリーを確認する必要があります。ノードで **crictl images** コマンドを使用するなど、イメージのプルソースを表示する他の方法では、イメージがミラーリングされた場所からプルされている場合でも、ミラーリングされていないイメージ名を表示します。



注記

Red Hat は、OpenShift Container Platform を使用してサードパーティーのレジストリーをテストしません。

関連情報

- CRI-O ログを表示してイメージソースを表示する方法の詳細は、Viewing the image pull source を参照してください。

4.4.4. 前提条件

- Red Hat Quay など、OpenShift Container Platform クラスターをホストする場所に Docker v2-2 をサポートするコンテナーイメージレジストリーを持っている。

注記

Red Hat Quay を使用する場合は、oc-mirror プラグインでバージョン 3.6 以降を使用する必要があります。Red Hat Quay のライセンスをお持ちの場合は、概念実証のため に、または Red Hat Quay Operator を使用 して Red Hat Quay をデプロイする方法を記載したドキュメントを参照してください。レジストリーの選択とインストールについてさらにサポートが必要な場合は、営業担当者または Red Hat サポートにお問い合わせください。

コンテナーイメージレジストリーの既存のソリューションがまだない場合には、OpenShift Container Platform のサブスクライバーに Red Hat Openshift 導入用のミラーレジストリー が提供されます。**Red Hat Openshift 導入用のミラーレジストリー**はサブスクリプションに含まれており、切断されたインストールで OpenShift Container Platform で必須のコンテナーイメージのミラーリングに使用できる小規模なコンテナーレジストリーです。

4.4.5. ミラーホストの準備

oc-mirror プラグインを使用してイメージをミラーリングする前に、プラグインをインストールし、コンテナーイメージレジストリーの認証情報ファイルを作成して、Red Hat からお使いのミラーへのミラーリングを許可する必要があります。

4.4.5.1. oc-mirror OpenShift CLI プラグインのインストール

oc-mirror OpenShift CLI プラグインをインストールして、切断された環境でイメージセットを管理します。

前提条件

- OpenShift CLI (**oc**) がインストールされている。完全な非接続環境でイメージセットをミラーリングする場合は、次の点を確認してください。
  - インターネットにアクセスできるホストに oc-mirror プラグインをインストールした。
  - 非接続環境のホストが、ターゲットミラーレジストリーにアクセスできる。
- oc-mirror を使用するオペレーティングシステムで、**umask** パラメーターを **0022** に設定した。
- 使用している RHEL バージョンに適したバイナリーをインストールした。

手順

1. oc-mirror CLI プラグインをダウンロードします。
   a. OpenShift Cluster Manager の ダウンロード ページに移動します。
   b. **OpenShift 切断インストールツール** セクションで、**OpenShift Client (oc) ミラープラグイン** の **ダウンロード** をクリックしてファイルを保存します。
2. アーカイブを抽出します。

```
$ tar xvzf oc-mirror.tar.gz
```

3. 必要に応じて、プラグインファイルを更新して実行可能にします。

```
$ chmod +x oc-mirror
```

注記

oc-mirror ファイルの名前を変更しないでください。

4. ファイルを **PATH** に配置して、oc-mirror CLI プラグインをインストールします (例: **/usr/local/bin**):。

```
$ sudo mv oc-mirror /usr/local/bin/.
```

検証

- 次のコマンドを実行して、oc-mirror v1 のプラグインが正常にインストールされたことを確認します。

```
$ oc mirror help
```

関連情報

- CLI プラグインのインストールおよび使用

4.4.5.2. イメージのミラーリングを可能にする認証情報の設定

Red Hat からミラーにイメージをミラーリングできるコンテナーイメージレジストリー認証情報ファイルを作成します。

警告

クラスターのインストール時に、このイメージレジストリー認証情報ファイルをプルシークレットとして使用しないでください。クラスターのインストール時にこのファイルを指定すると、クラスター内のすべてのマシンにミラーレジストリーへの書き込みアクセスが付与されます。

警告

このプロセスでは、ミラーレジストリーのコンテナーイメージレジストリーへの書き込みアクセスがあり、認証情報をレジストリープルシークレットに追加する必要があります。

前提条件

- 非接続環境で使用するミラーレジストリーを設定しました。
- イメージをミラーリングするミラーレジストリー上のイメージリポジトリーの場所を特定している。
- イメージのイメージリポジトリーへのアップロードを許可するミラーレジストリーアカウントをプロビジョニングしている。

手順

インストールホストで以下の手順を実行します。

1. **registry.redhat.io** プルシークレットを Red Hat OpenShift Cluster Manager からダウンロードします。
2. JSON 形式でプルシークレットのコピーを作成します。

```
$ cat ./pull-secret | jq . > <path>/<pull_secret_file_in_json> 1
```

1 プルシークレットを保存するフォルダーへのパスおよび作成する JSON ファイルの名前を指定します。

ファイルの内容は以下の例のようになります。

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
```

```
    "email": "you@example.com"
  },
  "quay.io": {
    "auth": "b3BlbnNo...",
    "email": "you@example.com"
  },
  "registry.connect.redhat.com": {
    "auth": "NTE3Njg5Nj...",
    "email": "you@example.com"
  },
  "registry.redhat.io": {
    "auth": "NTE3Njg5Nj...",
    "email": "you@example.com"
  }
 }
}
```

3. ファイルを ~/**.docker/config.json** または **$XDG_RUNTIME_DIR/containers/auth.json** として保存します。

1. ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードまたはトークンを生成します。

```
$ echo -n '<user_name>:<password>' | base64 -w0 (1)
BGVtbYk3ZHAtqXs=
```

(1) **<user_name>** および **<password>** については、レジストリーに設定したユーザー名およびパスワードを指定します。

2. JSON ファイルを編集し、レジストリーについて記述するセクションをこれに追加します。

```
"auths": {
  "<mirror_registry>": { (1)
    "auth": "<credentials>", (2)
    "email": "you@example.com"
  }
},
```

(1) ミラーレジストリーがコンテンツを提供するために使用するレジストリーのドメイン名およびポート (オプション) を指定します。例: **registry.example.com** または **registry.example.com:8443**

(2) ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

ファイルは以下の例のようになります。

```
{
 "auths": {
  "registry.example.com": {
    "auth": "BGVtbYk3ZHAtqXs=",
    "email": "you@example.com"
  },
```

```
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

4.4.6. イメージセット設定の作成

oc-mirror プラグインを使用してイメージセットをミラーリングする前に、イメージセット設定ファイルを作成する必要があります。このイメージセット設定ファイルは、ミラーリングする OpenShift Container Platform リリース、Operator、およびその他のイメージと、oc-mirror プラグインの他の設定を定義します。

イメージセット設定ファイルでストレージバックエンドを指定する必要があります。このストレージバックエンドは、 Docker v2-2 をサポートするローカルディレクトリーまたはレジストリーにすることができます。oc-mirror プラグインは、イメージセットの作成中にこのストレージバックエンドにメタデータを保存します。

重要

oc-mirror プラグインによって生成されたメタデータを削除または変更しないでください。同じミラーレジストリーに対して oc-mirror プラグインを実行するたびに、同じストレージバックエンドを使用する必要があります。

前提条件

- コンテナーイメージレジストリーの認証情報ファイルを作成している。手順については、イメージのミラーリングを可能にする認証情報の設定 を参照してください。

手順

1. **oc mirror init** コマンドを使用して、イメージセット設定のテンプレートを作成し、それを **imageset-config.yaml** というファイルに保存します。

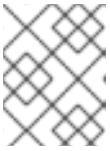
```
$ oc mirror init <--registry <storage_backend> > imageset-config.yaml 1
```

1 ストレージバックエンドのロケーションを指定します (例: **example.com/mirror/oc-mirror-metadata**)。

2. ファイルを編集し、必要に応じて設定を調整します。

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
archiveSize: 4 (1)
storageConfig: (2)
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata (3)
    skipTLS: false
mirror:
  platform:
    channels:
    - name: stable-4.16 (4)
      type: ocp
    graph: true (5)
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16 (6)
    packages:
    - name: serverless-operator (7)
      channels:
      - name: stable (8)
  additionalImages:
  - name: registry.redhat.io/ubi9/ubi:latest (9)
  helm: {}
```

1. **archiveSize** を追加して、イメージセット内の各ファイルの最大サイズを GiB 単位で設定します。
2. イメージセットのメタデータを保存するバックエンドの場所を設定します。この場所は、レジストリーまたはローカルディレクトリーにすることができます。**storageConfig**値を指定する必要があります。
3. ストレージバックエンドのレジストリー URL を設定します。
4. OpenShift Container Platform イメージを取得するためのチャネルを設定します。
5. **graph: true** を追加して、グラフデータイメージをビルドし、ミラーレジストリーにプッシュします。OpenShift Update Service (OSUS) を作成するには、graph-data イメージが必要です。**graph: true** フィールドは **UpdateService** カスタムリソースマニフェストも生成します。**oc** コマンドラインインターフェイス (CLI) は、**UpdateService** カスタムリソースマニフェストを使用して OSUS を作成できます。詳細については、**OpenShift Update Service について** を参照してください。
6. OpenShift Container Platform イメージを取得するための Operator カタログを設定します。
7. イメージセットに含める特定の Operator パッケージのみを指定します。カタログ内のすべてのパッケージを取得するには、このフィールドを削除してください。
8. イメージセットに含める Operator パッケージの特定のチャネルのみを指定します。そのチャネルでバンドルを使用しない場合も、常に Operator パッケージのデフォルトチャネルを含める必要があります。コマンド **oc mirror list operators --catalog=<catalog_name> --package=<package_name>** を実行すると、デフォルトチャネルを見つけることができます。
9. イメージセットに含める追加のイメージを指定します。



注記

graph: true フィールド は、他のミラーリングされたイメージとともに **ubi-micro** イメージもミラーリングします。

パラメーターの完全なリストについては、イメージセットの設定パラメーター を参照してください。また、さまざまなミラーリングのユースケースについては、イメージセットの設定例 を参照してください。

3. 更新したファイルを保存します。
このイメージセット設定ファイルは、コンテンツをミラーリングするときに **oc mirror** コマンドで必要になります。

関連情報

- Image set configuration parameters
- Image set configuration examples
- 非接続環境での OpenShift Update Service の使用

4.4.7. イメージセットをミラーレジストリーにミラーリングする

oc-mirror CLI プラグインを使用して、部分的な非接続環境 または 完全な非接続環境 でイメージをミラーレジストリーにミラーリングできます。

これらの手順は、ミラーレジストリーがすでに設定されていることを前提としています。

4.4.7.1. 部分的な非接続環境でのイメージセットのミラーリング

部分的な非接続環境では、イメージセットをターゲットミラーレジストリーに直接ミラーリングできます。

4.4.7.1.1. ミラーからミラーへのミラーリング

oc-mirror プラグインを使用して、イメージセットの作成中にアクセス可能なターゲットミラーレジストリーにイメージセットを直接ミラーリングできます。

イメージセット設定ファイルでストレージバックエンドを指定する必要があります。このストレージバックエンドは、ローカルディレクトリーまたは Dockerv2 レジストリーにすることができます。oc-mirror プラグインは、イメージセットの作成中にこのストレージバックエンドにメタデータを保存します。

重要

oc-mirror プラグインによって生成されたメタデータを削除または変更しないでください。同じミラーレジストリーに対して oc-mirror プラグインを実行するたびに、同じストレージバックエンドを使用する必要があります。

前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。

- **oc-mirror** CLI プラグインをインストールしている。
- イメージセット設定ファイルを作成している。

手順

- **oc mirror** コマンドを実行して、指定されたイメージセット設定から指定されたレジストリーにイメージをミラーリングします。

```
$ oc mirror --config=./<imageset-config.yaml> \ 1
  docker://registry.example:5000                2
```

1. 作成したイメージセット設定ファイルを指定します。たとえば、**imageset-config.yaml** です。
2. イメージセットファイルをミラーリングするレジストリーを指定します。レジストリーは **docker://** で始まる必要があります。ミラーレジストリーに最上位の namespace を指定する場合は、これ以降の実行でもこれと同じ namespace を使用する必要があります。

検証

1. 生成された **oc-mirror-workspace/** ディレクトリーに移動します。
2. results ディレクトリーに移動します (例: **results-1639608409/**。
3. **ImageContentSourcePolicy** および **CatalogSource** リソースに YAML ファイルが存在することを確認します。

注記

ImageContentSourcePolicy YAML ファイルの **repositoryDigestMirrors** セクションは、インストール中に **install-config.yaml** ファイルとして使用されます。

次のステップ

- oc-mirror が生成したリソースを使用するようにクラスターを設定します。

トラブルシューティング

- Unable to retrieve source image .

4.4.7.2. 完全な非接続環境でのイメージセットのミラーリング

完全な非接続環境でイメージセットをミラーリングするには、最初に イメージセットをディスクにミラーリング してから、ディスク上のイメージセットファイルをミラーにミラーリング する必要があります。

4.4.7.2.1. ミラーからディスクへのミラーリング

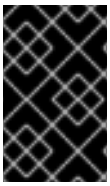
oc-mirror プラグインを使用して、イメージセットを生成し、コンテンツをディスクに保存できます。生成されたイメージセットは、非接続環境に転送され、ターゲットレジストリーにミラーリングされます。

重要

イメージセット設定ファイルで指定されている設定によっては、oc-mirror を使用してイメージをミラーリングすると、数百ギガバイトのデータがディスクにダウンロードされる場合があります。

多くの場合、ミラーレジストリーにデータを入力するときの最初のイメージセットのダウンロードが、最も大きなものとなります。最後にコマンドを実行した後に変更されたイメージのみをダウンロードするため、oc-mirror プラグインを再度実行すると、生成されるイメージセットは小さいことが多いです。

イメージセット設定ファイルでストレージバックエンドを指定する必要があります。このストレージバックエンドは、ローカルディレクトリーまたは docker v2 レジストリーにすることができます。oc-mirror プラグインは、イメージセットの作成中にこのストレージバックエンドにメタデータを保存します。



重要

oc-mirror プラグインによって生成されたメタデータを削除または変更しないでください。同じミラーレジストリーに対して oc-mirror プラグインを実行するたびに、同じストレージバックエンドを使用する必要があります。

前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。
- **oc-mirror** CLI プラグインをインストールしている。
- イメージセット設定ファイルを作成している。

手順

- **oc mirror** コマンドを実行して、指定されたイメージセット設定からディスクにイメージをミラーリングします。

```
$ oc mirror --config=./imageset-config.yaml \ 1
  file://<path_to_output_directory>           2
```

1. 作成されたイメージセット設定ファイルを渡します。この手順では、**imageset-config.yaml** という名前であることを前提としています。
2. イメージセットファイルを出力するターゲットディレクトリーを指定します。ターゲットディレクトリーのパスは、**file://** で始まる必要があります。

検証

1. 出力ディレクトリーに移動します。

```
$ cd <path_to_output_directory>
```

2. イメージセットの **.tar** ファイルが作成されたことを確認します。

```
$ ls
```

出力例

```
mirror_seq1_000000.tar
```

次のステップ

- イメージセットの.tar ファイルを非接続環境に転送します。

トラブルシューティング

- Unable to retrieve source image .

4.4.7.2.2. ディスクからミラーへのミラーリング

oc-mirror プラグインを使用して、生成されたイメージセットの内容をターゲットミラーレジストリーにミラーリングできます。

前提条件

- 非接続環境に OpenShift CLI (**oc**) をインストールしている。
- 非接続環境に **oc-mirror** CLI プラグインをインストールしている。
- **ocmirror** コマンドを使用してイメージセットファイルを生成している。
- イメージセットファイルを非接続環境に転送しました。

手順

- **oc mirror** コマンドを実行して、ディスク上のイメージセットファイルを処理し、その内容をターゲットミラーレジストリーにミラーリングします。

```
$ oc mirror --from=./mirror_seq1_000000.tar \ 1
  docker://registry.example:5000              2
```

1. この例では、**mirror_seq1_000000.tar** という名前のイメージセット.tar ファイルをミラーに渡します。イメージセット設定ファイルで **archiveSize** 値が指定されている場合、イメージセットは複数の.tar ファイルに分割される可能性があります。この状況では、イメージセットの.tar ファイルを含むディレクトリーを渡すことができます。
2. イメージセットファイルをミラーリングするレジストリーを指定します。レジストリーは **docker://** で始まる必要があります。ミラーレジストリーに最上位の namespace を指定する場合は、これ以降の実行でもこれと同じ namespace を使用する必要があります。

このコマンドは、ミラーレジストリーをイメージセットで更新し、**ImageContentSourcePolicy** および **CatalogSource** リソースを生成します。

検証

1. 生成された **oc-mirror-workspace/** ディレクトリーに移動します。

2. results ディレクトリーに移動します (例: **results-1639608409**/。
3. **ImageContentSourcePolicy** および **CatalogSource** リソースに YAML ファイルが存在することを確認します。

次のステップ

- oc-mirror が生成したリソースを使用するようにクラスターを設定します。

トラブルシューティング

- Unable to retrieve source image .

4.4.8. oc-mirror が生成したリソースを使用するためのクラスター設定

イメージセットをミラーレジストリーにミラーリングした後に、生成された **ImageContentSourcePolicy**、**CatalogSource**、およびリリースイメージの署名リソースをクラスターに適用する必要があります。

ImageContentSourcePolicy リソースは、ミラーレジストリーをソースレジストリーに関連付け、イメージプル要求をオンラインレジストリーからミラーレジストリーにリダイレクトします。**CatalogSource** リソースは、Operator Lifecycle Manager (OLM) によって使用され、ミラーレジストリーで使用可能な Operator に関する情報を取得します。リリースイメージの署名は、ミラーリングされたリリースイメージの検証に使用されます。

前提条件

- 非接続環境で、イメージセットをレジストリーミラーにミラーリングしました。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift CLI にログインします。
2. 以下のコマンドを実行して、results ディレクトリーからクラスターに YAML ファイルを適用します。

```
$ oc apply -f ./oc-mirror-workspace/results-1639608409/
```

3. リリースイメージをミラーリングした場合は、次のコマンドを実行して、リリースイメージの署名をクラスターに適用します。

```
$ oc apply -f ./oc-mirror-workspace/results-1639608409/release-signatures/
```

注記

クラスターではなく Operator をミラーリングしている場合、**$ oc apply -f ./oc-mirror-workspace/results-1639608409/release-signatures/** を実行する必要はありません。適用するリリースイメージ署名がないため、このコマンドを実行するとエラーが返されます。

検証

1. 以下のコマンドを実行して、**ImageContentSourcePolicy** リソースが正常にインストールされたことを確認します。

```
$ oc get imagecontentsourcepolicy
```

2. 以下のコマンドを実行して、**CatalogSource** リソースが正常にインストールされたことを確認します。

```
$ oc get catalogsource -n openshift-marketplace
```

4.4.9. ミラーレジストリーコンテンツの更新

イメージセット設定ファイルを更新し、イメージセットをミラーレジストリーにミラーリングすることで、ミラーレジストリーの内容を更新できます。oc-mirror プラグインを次回実行したときに、前回の実行以降の新規イメージと更新されたイメージのみを含むイメージセットが生成されます。

ミラーレジストリーを更新する際には、次の点を考慮する必要があります。

- 生成およびミラーリングされた最新のイメージセットにイメージが含まれていない場合、そのイメージはターゲットミラーレジストリーからプルーニングされます。したがって、差分イメージセットのみが作成およびミラーリングされるように、以下と同じ組み合わせの主要コンポーネントのイメージを更新してください。
  - イメージセットの設定
  - 宛先レジストリー
  - ストレージの設定
- ディスクからミラー、またはミラーからミラーへのワークフローの場合、イメージがプルーニングされる可能性があります。
- 生成されたイメージセットは、ターゲットミラーレジストリーに順番にプッシュする必要があります。シーケンス番号は、生成されたイメージセットアーカイブファイルのファイル名から取得できます。
- oc-mirror プラグインによって生成されたメタデータイメージを削除または変更しないでください。
- イメージセットの最初の作成時にミラーレジストリーにトップレベルの namespace を指定した場合は、同じミラーレジストリーに対して oc-mirror プラグインを実行するたびに、この同じ namespace を使用する必要があります。

ミラーレジストリーコンテンツを更新するワークフローの詳細は、「ワークフローの概要」セクションを参照してください。

4.4.9.1. ミラーレジストリーの更新例

このセクションでは、ミラーレジストリーをディスクからミラーに更新するユースケースについて説明します。

以前ミラーリングに使用した ImageSetConfiguration ファイルの例:

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
```

```
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.12.1
        maxVersion: 4.12.1
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
        - name: rhacs-operator
          channels:
          - name: stable
```

既存のイメージをプルーニングして特定の OpenShift Container Platform バージョンをミラーリングする

更新された ImageSetConfiguration ファイル:

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.13 (1)
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
        - name: rhacs-operator
          channels:
          - name: stable
```

(1) **stable-4.13** に置き換えると、**stable-4.12** のすべてのイメージがプルーニングされます。

既存のイメージをプルーニングして Operator の最新バージョンに更新する

更新された ImageSetConfiguration ファイル:

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.12.1
        maxVersion: 4.12.1
```

```
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
        - name: rhacs-operator
          channels:
          - name: stable 1
```

1 バージョンを指定せずに同じチャネルを使用すると、既存のイメージがプルーニングされ、最新バージョンのイメージに更新されます。

既存の Operator をプルーニングして新しい Operator をミラーリングする

更新された ImageSetConfiguration ファイル:

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.12.1
        maxVersion: 4.12.1
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
        - name: <new_operator_name> 1
          channels:
          - name: stable
```

1 **rhacs-operator** を **new_operator_name** に置き換えると、Red Hat Advanced Cluster Security for Kubernetes Operator がプルーニングされます。

すべての OpenShift Container Platform イメージをプルーニングする

更新された ImageSetConfiguration ファイル:

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
```

関連情報

- Image set configuration examples
- 部分的な非接続環境でのイメージセットのミラーリング
- 完全な非接続環境でのイメージセットのミラーリング
- oc-mirror が生成したリソースを使用するためのクラスター設定

4.4.10. ドライランの実行

実際にイメージをミラーリングせずに、oc-mirror を使用してドライランを実行できます。これにより、ミラーリングされるイメージのリストと、ミラーレジストリーからプルーニングされるイメージを確認できます。ドライランを使用すると、イメージセット設定のエラーを早期に検出したり、生成されたイメージリストを他のツールで使用してミラーリング操作を実行したりすることもできます。

前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。
- **oc-mirror** CLI プラグインをインストールしている。
- イメージセット設定ファイルを作成している。

手順

1. **--dry-run** フラグを指定して **oc mirror** コマンドを実行し、ドライランを実行します。

```
$ oc mirror --config=./imageset-config.yaml \ 1
  docker://registry.example:5000           \ 2
  --dry-run                                  3
```

1. 作成されたイメージセット設定ファイルを渡します。この手順では、**imageset-config.yaml** という名前であることを前提としています。
2. ミラーレジストリーを指定します。**--dry-run** フラグを使用している限り、このレジストリーには何もミラーリングされません。
3. **--dry-run** フラグを使用して、実際のイメージセットファイルではなく、ドライランアーティファクトを生成します。

出力例

```
Checking push permissions for registry.example:5000
Creating directory: oc-mirror-workspace/src/publish
Creating directory: oc-mirror-workspace/src/v2
Creating directory: oc-mirror-workspace/src/charts
Creating directory: oc-mirror-workspace/src/release-signatures
No metadata detected, creating new workspace
wrote mirroring manifests to oc-mirror-workspace/operators.1658342351/manifests-redhat-operator-index

...
```

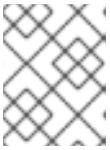
```
info: Planning completed in 31.48s
info: Dry run complete
Writing image mapping to oc-mirror-workspace/mapping.txt
```

2. 生成されたワークスペースディレクトリーに移動します。

```
$ cd oc-mirror-workspace/
```

3. 生成された **mapping.txt** ファイルを確認します。
このファイルには、ミラーリングされるすべてのイメージのリストが含まれています。

4. 生成された **pruning-plan.json** ファイルを確認します。
このファイルには、イメージセットの公開時にミラーレジストリーからプルーニングされるすべてのイメージのリストが含まれています。



注記

pruning-plan.json ファイルは、oc-mirror コマンドがミラーレジストリーを指し、プルーニングするイメージがある場合にのみ生成されます。

4.4.11. ローカルの OCI Operator カタログを含む

OpenShift Container Platform リリース、Operator カタログ、および追加イメージをレジストリーから部分的に切断されたクラスターにミラーリングするときに、ディスク上のローカルのファイルベースのカタログから Operator カタログイメージを含めることができます。ローカルカタログは Open Container Initiative (OCI) 形式である必要があります。

ローカルカタログとそのコンテンツは、イメージセット設定ファイル内のフィルタリング情報に基づいて、ターゲットミラーレジストリーにミラーリングされます。

重要

ローカル OCI カタログをミラーリングする場合、ローカル OCI 形式のカタログとともにミラーリングする OpenShift Container Platform リリースまたは追加のイメージをレジストリーからプルする必要があります。

OCI カタログを oc-mirror イメージセットファイルと一緒にディスク上でミラーリングすることはできません。

OCI 機能を使用するユースケースの 1 つの例は、ディスク上の場所に OCI カタログを構築している CI/CD システムがあり、その OCI カタログを OpenShift Container Platform リリースとともにミラーレジストリーにミラーリングしたい場合です。

注記

OpenShift Container Platform 4.12 の oc-mirror プラグインの Technology Preview OCI ローカルカタログ機能を使用した場合、完全に切断されたクラスターへのミラーリングの最初のステップとして、ローカルにカタログをコピーして OCI 形式に変換するために oc-mirror プラグインの OCI ローカルカタログ機能を使用できないようになりました。

前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。
- **oc-mirror** CLI プラグインをインストールしている。

手順

1. イメージセット設定ファイルを作成し、必要に応じて設定を調整します。
次のイメージセット設定例では、OpenShift Container Platform リリースおよび **registry.redhat.io** の UBI イメージとともに、ディスク上の OCI カタログをミラーリングします。

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
storageConfig:
  local:
    path: /home/user/metadata                                    1
mirror:
  platform:
    channels:
    - name: stable-4.16                                          2
      type: ocp
    graph: false
  operators:
  - catalog: oci:///home/user/oc-mirror/my-oci-catalog           3
    targetCatalog: my-namespace/redhat-operator-index            4
    packages:
    - name: aws-load-balancer-operator
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16   5
    packages:
    - name: rhacs-operator
  additionalImages:
  - name: registry.redhat.io/ubi9/ubi:latest                     6
```

1. イメージセットのメタデータを保存するバックエンドの場所を設定します。この場所は、レジストリーまたはローカルディレクトリーにすることができます。**storageConfig**値を指定する必要があります。
2. オプションで、**registry.redhat.io** からミラーリングする OpenShift Container Platform リリースを含めます。
3. ディスク上の OCI カタログの場所への絶対パスを指定します。OCI 機能を使用する場合、パスは **oci:**// で始まる必要があります。
4. 必要に応じて、カタログをミラーリングする代替の namespace と名前を指定します。
5. 必要に応じて、レジストリーから取得する追加の Operator カタログを指定します。
6. 必要に応じて、レジストリーからプルする追加のイメージを指定します。

2. **oc mirror** コマンドを実行して、OCI カタログをターゲットミラーレジストリーにミラーリングします。

```
$ oc mirror --config=./imageset-config.yaml \ 1
  docker://registry.example:5000              2
```

1 イメージセット設定ファイルを渡します。この手順では、**imageset-config.yaml** という名前であることを前提としています。

2 コンテンツをミラーリングするレジストリーを指定します。レジストリーは **docker://** で始まる必要があります。ミラーレジストリーに最上位の namespace を指定する場合は、これ以降の実行でもこれと同じ namespace を使用する必要があります。

オプションで、他のフラグを指定して OCI 機能の動作を調整できます。

--oci-insecure-signature-policy

署名をターゲットミラーレジストリーにプッシュしないでください。

--oci-registries-config

TOML 形式の **registries.conf** ファイルへのパスを指定します。これを使用して、イメージセット設定ファイルを変更することなく、テスト用の運用前の場所など、別のレジストリーからミラーリングできます。このフラグはローカル OCI カタログにのみ影響し、他のミラーリングされたコンテンツには影響しません。

registries.conf ファイルの例

```
[[registry]]
 location = "registry.redhat.io:5000"
 insecure = false
 blocked = false
 mirror-by-digest-only = true
 prefix = ""
 [[registry.mirror]]
   location = "preprod-registry.example.com"
   insecure = false
```

次のステップ

- oc-mirror が生成したリソースを使用するようにクラスターを設定します。

関連情報

- oc-mirror が生成したリソースを使用するためのクラスター設定

4.4.12. Image set configuration parameters

oc-mirror プラグインには、ミラーリングするイメージを定義するイメージセット設定ファイルが必要です。次の表に、**ImageSetConfiguration** リソースで使用可能なパラメーターを示します。

表4.1 ImageSetConfiguration パラメーター

パラメーター	説明	値

パラメーター	説明	値
apiVersion	**ImageSetConfiguration** コンテンツの API バージョン。	文字列。例: **mirror.openshift.io/v1alpha2**
archiveSize	イメージセット内の各アーカイブファイルの最大サイズ (GiB 単位)。	integer例: **4**
mirror	イメージセットの設定。	オブジェクト
mirror.additionalImages	イメージセットの追加のイメージ設定。	オブジェクトの配列。以下に例を示します。 additionalImages: - name: registry.redhat.io/ubi8/ubi:latest
mirror.additionalImages.name	ミラーリングするイメージのタグまたはダイジェスト。	文字列。例: **registry.redhat.io/ubi8/ubi:latest**
mirror.blockedImages	ミラーリングからブロックするイメージの完全なタグ、ダイジェスト、またはパターン。	文字列の配列例: **docker.io/library/alpine**
mirror.helm	イメージセットのヘルム設定。oc-mirror プラグインは、レンダリング時にユーザー入力を必要としないヘルムチャートのみをサポートすることに注意してください。	オブジェクト
mirror.helm.local	ミラーリングするローカルヘルムチャート。	オブジェクトの配列。以下に例を示します。 local: - name: podinfo path: /test/podinfo-5.0.0.tar.gz

パラメーター	説明	値
mirror.helm.local.name	ミラーリングするローカルヘルムチャートの名前。	文字列。例: **podinfo**。
mirror.helm.local.path	ミラーリングするローカルヘルムチャートのパス。	文字列。例: **/test/podinfo-5.0.0.tar.gz**
mirror.helm.repositories	ミラーリング元のリモートヘルムリポジトリー。	オブジェクトの配列。以下に例を示します。 repositories: - name: podinfo url: https://example.github.io/podinfo charts: - name: podinfo version: 5.0.0
mirror.helm.repositories.name	ミラーリング元のヘルムリポジトリーの名前。	文字列。例: **podinfo**。
mirror.helm.repositories.url	ミラーリング元の helm リポジトリーの URL。	文字列。例: **https://example.github.io/podinfo**
mirror.helm.repositories.charts	ミラーリングするリモートヘルムチャート。	オブジェクトの配列。
mirror.helm.repositories.charts.name	ミラーリングするヘルムチャートの名前。	文字列。例: **podinfo**。
mirror.helm.repositories.charts.version	ミラーリングする名前付きヘルムチャートのバージョン。	文字列。例: **5.0.0**。

パラメーター	説明	値
mirror.operators	イメージセットの Operators 設定。	オブジェクトの配列。以下に例を示します。 operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16 packages: - name: elasticsearch-operator minVersion: '2.4.0'
mirror.operators.catalog	イメージセットに含める Operator カタログ。	文字列。たとえば、**registry.redhat.io/redhat/redhat-operator-index:v4.15** です。
mirror.operators.full	**true** の場合、完全なカタログ、Operator パッケージ、または Operator チャネルをダウンロードします。	ブール値。デフォルト値は **false** です。
mirror.operators.packages	Operator パッケージ設定	オブジェクトの配列。以下に例を示します。 operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16 packages: - name: elasticsearch-operator minVersion: '5.2.3-31'

パラメーター	説明	値
mirror.operators.packages.name	イメージセットに含める Operator パッケージ名	文字列。例: **elasticsearch-operator**
mirror.operators.packages.channels	Operator パッケージのチャネル設定。	オブジェクト
mirror.operators.packages.channels.name	イメージセットに含める、パッケージ内で一意の Operator チャネル名。	文字列。たとえば、**fast** または **stable-v4.15** です。
mirror.operators.packages.channels.maxVersion	Operator が存在するすべてのチャネルでミラーリングする最上位バージョンの Operator。詳細は、以下の注記を参照してください。	文字列。例: **5.2.3-31**
mirror.operators.packages.channels.minBundle	含める最小バンドルの名前と、チャネルヘッドへの更新グラフ内のすべてのバンドル。名前付きバンドルにセマンティックバージョンメタデータがない場合にのみ、このフィールドを設定します。	文字列。例：**bundleName**
mirror.operators.packages.channels.minVersion	存在するすべてのチャネル間でミラーリングする Operator の最低バージョン。詳細は、以下の注記を参照してください。	文字列。例: **5.2.3-31**
mirror.operators.packages.maxVersion	Operator が存在するすべてのチャネルでミラーリングする最上位バージョンの Operator。詳細は、以下の注記を参照してください。	文字列。例: **5.2.3-31**。
mirror.operators.packages.minVersion	存在するすべてのチャネル間でミラーリングする Operator の最低バージョン。詳細は、以下の注記を参照してください。	文字列。例: **5.2.3-31**。
mirror.operators.skipDependencies	**true** の場合、バンドルの依存関係は含まれません。	ブール値。デフォルト値は **false** です。
mirror.operators.targetCatalog	参照されるカタログをミラーリングするための代替名とオプションの namespace 階層。	文字列。例: **my-namespace/my-operator-catalog**

パラメーター	説明	値
mirror.operators.targetName	参照されたカタログをミラーリングするための代替名。 **targetName** パラメーターは非推奨になりました。代わりに **targetCatalog** パラメーターを使用してください。	文字列。例： **my-operator-catalog**
mirror.operators.targetTag	**targetName** または **targetCatalog** に追加する代替タグ。	文字列。例: **v1**
mirror.platform	イメージセットのプラットフォーム設定。	オブジェクト
mirror.platform.architectures	ミラーリングするプラットフォームリリースペイロードのアーキテクチャー。	文字列の配列以下に例を示します。 architectures: - amd64 - arm64 - multi - ppc64le - s390x デフォルト値は **amd64** です。値 **multi** を指定すると、使用可能なすべてのアーキテクチャーでミラーリングがサポートされ、個別のアーキテクチャーを指定する必要がなくなります。
mirror.platform.channels	イメージセットのプラットフォームチャネル設定。	オブジェクトの配列。以下に例を示します。 channels: - name: stable-4.10 - name: stable-4.16

パラメーター	説明	値
mirror.platform.channels.full	**true** の場合、**minVersion** をチャネルの最初のリリースに設定し、**maxVersion** をチャネルの最後のリリースに設定します。	ブール値。デフォルト値は **false** です。
mirror.platform.channels.name	リリースチャネルの名前。	文字列。たとえば、**stable-4.16** などです。
mirror.platform.channels.minVersion	ミラーリングされる参照プラットフォームの最小バージョン。	文字列。例: **4.12.6**
mirror.platform.channels.maxVersion	ミラーリングされる参照プラットフォームの最上位バージョン。	文字列。例: **10.0.0.0/24** など。
mirror.platform.channels.shortestPath	最短パスミラーリングまたはフルレンジミラーリングを切り替えます。	ブール値。デフォルト値は **false** です。
mirror.platform.channels.type	ミラーリングするプラットフォームのタイプ。	文字列。例： **ocp** または **okd**。デフォルトは **ocp** です。
mirror.platform.graph	OSUS グラフがイメージセットに追加され、その後ミラーに公開されるかどうかを示します。	ブール値。デフォルト値は **false** です。
storageConfig	イメージセットのバックエンド設定。	オブジェクト
storageConfig.local	イメージセットのローカルバックエンド設定。	オブジェクト
storageConfig.local.path	イメージセットのメタデータを含むディレクトリーのパス。	文字列。例: **./path/to/dir/**
storageConfig.registry	イメージセットのレジストリーバックエンド設定。	オブジェクト
storageConfig.registry.imageURL	バックエンドレジストリー URI。オプションで、URI に namespace 参照を含めることができます。	文字列。例: **quay.io/myuser/imageset:metadata**
storageConfig.registry.skipTLS	オプションで、参照されるバックエンドレジストリーの TLS 検証をスキップします。	ブール値。デフォルト値は **false** です。

注記

minVersion および **maxVersion** プロパティーを使用して特定の Operator バージョン範囲をフィルターすると、複数のチャネルヘッドエラーが発生する可能性があります。エラーメッセージには、**multiple channel heads** があることが示されます。これは、フィルターを適用すると、Operator の更新グラフが切り捨てられるためです。

すべての Operator チャネルに、1 つのエンドポイント (つまり最新バージョンの Operator) を持つ更新グラフを構成するバージョンが含まれている必要があります。これは Operator Lifecycle Manager によって要求される要件です。フィルター範囲を適用すると、更新グラフが 2 つ以上の個別のグラフ、または複数のエンドポイントを持つグラフに変換されることがあります。

このエラーを回避するには、最新バージョンの Operator を除外しないでください。それでもエラーが発生する場合は、Operator に応じて、**maxVersion** プロパティーを増やすか、**minVersion** プロパティーを減らす必要があります。Operator グラフはそれぞれ異なる可能性があるため、エラーが解決するまでこれらの値を調整する必要がある場合があります。

4.4.13. Image set configuration examples

次の **ImageSetConfiguration** ファイルの例は、さまざまなミラーリングのユースケースの設定を示しています。

ユースケース: 最短の OpenShift Container Platform 更新パスを含める
以下の **ImageSetConfiguration** ファイルは、ローカルストレージバックエンドを使用し、最小バージョン **4.11.37** から最大バージョン **4.12.15** への最短更新パスに沿ってすべての OpenShift Container Platform バージョンを含めます。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.11.37
        maxVersion: 4.12.15
        shortestPath: true
```

使用事例: マルチアーキテクチャーリリースの最小バージョンから最新バージョンまでの OpenShift Container Platform のすべてのバージョンを含める
以下の **ImageSetConfiguration** ファイルは、レジストリーストレージバックエンドを使用し、最小バージョン **4.13.4** からチャネルの最新バージョンまでのすべての OpenShift Container Platform バージョンを含みます。このイメージセット設定で oc-mirror を呼び出すたびに、**stable-4.13** チャネルの最新リリースが評価されるため、定期的に oc-mirror を実行すると、OpenShift Container Platform イメージの最新リリースを自動的に受け取ることができます。

platform.architectures の値を **multi** に設定すると、マルチアーキテクチャーリリースでミラーリングがサポートされるようになります。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  platform:
    architectures:
      - "multi"
    channels:
      - name: stable-4.13
        minVersion: 4.13.4
        maxVersion: 4.13.6
```

ユースケース: 最小から最新までの Operator バージョンを含める
次の**ImageSetConfiguration**ファイルは、ローカルストレージバックエンドを使用し、これには、**stable** チャネルの Kubernetes Operator 用の Red Hat Advanced Cluster Security (4.0.1 以降のバージョン) のみが含まれています。

注記

最小または最大のバージョン範囲を指定した場合、その範囲内のすべての Operator バージョンを受信できない可能性があります。

デフォルトで、oc-mirror は、Operator Lifecycle Manager (OLM) 仕様でスキップされたバージョン、または新しいバージョンに置き換えられたバージョンを除外します。スキップされた Operator のバージョンは、CVE の影響を受けるか、バグが含まれている可能性があります。代わりに新しいバージョンを使用してください。スキップおよび置き換えられたバージョンの詳細は、OLM を使用した更新グラフの作成 を参照してください。

指定した範囲内のすべての Operator バージョンを受信するには、**mirror.operators.full** フィールドを **true** に設定します。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16
      packages:
        - name: rhacs-operator
          channels:
          - name: stable
            minVersion: 4.0.1
```

注記

最新バージョンではなく最大バージョンを指定するに
は、**mirror.operators.packages.channels.maxVersion** フィールドを設定します。

ユースケース: Nutanix CSI Operator を含める
次の **ImageSetConfiguration** ファイルは、ローカルストレージバックエンドを使用します。このファイルには、Nutanix CSI Operator、OpenShift Update Service (OSUS) グラフイメージ、および追加の Red Hat Universal Base Image (UBI) が含まれます。

ImageSetConfigurationファイルの例

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
storageConfig:
  registry:
    imageURL: mylocalregistry/ocp-mirror/openshift4
    skipTLS: false
mirror:
  platform:
    channels:
    - name: stable-4.11
      type: ocp
    graph: true
  operators:
  - catalog: registry.redhat.io/redhat/certified-operator-index:v4.16
    packages:
    - name: nutanixcsioperator
      channels:
      - name: stable
  additionalImages:
  - name: registry.redhat.io/ubi9/ubi:latest
```

ユースケース: デフォルトの Operator チャネルを含める
次の **ImageSetConfiguration** ファイルには、OpenShift Elasticsearch Operator の **stable-5.7** および **stable** チャネルが含まれています。**安定版 5.7** チャネルのパッケージのみが必要な場合でも、**stable** チャネルは Operator のデフォルトチャネルであるため、**ImageSetConfiguration** ファイルにも含める必要があります。そのチャネルでバンドルを使用しない場合も、常に Operator パッケージのデフォルトチャネルを含める必要があります。

ヒント

コマンド **oc mirror list operators --catalog=<catalog_name> --package=<package_name>** を実行すると、デフォルトチャネルを見つけることができます。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
```

```
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16
    packages:
    - name: elasticsearch-operator
      channels:
      - name: stable-5.7
      - name: stable
```

ユースケース: カタログ全体を含める (すべてのバージョン)
次の **ImageSetConfiguration** ファイルは、**mirror.operators.full** フィールドを **true** に設定して、Operator カタログ全体のすべてのバージョンを含めます。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16
      full: true
```

ユースケース: カタログ全体を含める (チャネルヘッドのみ)
次の **ImageSetConfiguration** ファイルには、Operator カタログ全体のチャネルヘッドが含まれています。

デフォルトでは、カタログ内の各 Operator において、oc-mirror にはデフォルトチャネルから Operator の最新バージョン (チャネルヘッド) が含まれています。チャネルヘッドだけでなく、すべての Operator バージョンをミラーリングする場合は、**mirror.operators.full** フィールドを **true** に設定する必要があります。

この例では、**targetCatalog** フィールドを使用して、カタログをミラーリングする代替 namespace と名前も指定します。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16
    targetCatalog: my-namespace/my-operator-catalog
```

ユースケース: 任意のイメージとヘルムチャートを含む
次の **ImageSetConfiguration** ファイルは、レジストリーストレージバックエンドを使用し、これにはヘルムチャートと追加の Red Hat Universal Base Image (UBI) が含まれています。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
archiveSize: 4
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
 platform:
   architectures:
     - "s390x"
   channels:
     - name: stable-4.16
 operators:
   - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16
 helm:
   repositories:
     - name: redhat-helm-charts
       url: https://raw.githubusercontent.com/redhat-developer/redhat-helm-charts/master
       charts:
         - name: ibm-mongodb-enterprise-helm
           version: 0.2.0
 additionalImages:
   - name: registry.redhat.io/ubi9/ubi:latest
```

4.4.14. oc-mirror のコマンドリファレンス

以下の表は、**oc mirror** サブコマンドとフラグについて説明しています。

表4.2 oc mirror サブコマンド

サブコマンド	説明
completion	指定されたシェルのオートコンプリートスクリプトを生成します。
describe	イメージセットの内容を出力します。
help	サブコマンドに関するヘルプを表示します。
init	初期イメージセット設定テンプレートを出力します。
list	利用可能なプラットフォームと Operator のコンテンツとそのバージョンを一覧表示します。
version	oc-mirror バージョンを出力します。

表4.3 oc mirror フラグ

フラグ	説明
-c, **--config <string>**	イメージセット設定ファイルへのパスを指定します。

フラグ	説明
--continue-on-error	イメージのプルに関連しないエラーが発生した場合は、続行して、可能な限りミラーリングを試みます。
--dest-skip-tls	ターゲットレジストリーの TLS 検証を無効にします。
--dest-use-http	ターゲットレジストリーにはプレーン HTTP を使用します。
--dry-run	イメージをミラーリングせずにアクションを出力します。**mapping.txt** ファイルおよび **pruning-plan.json** ファイルを生成します。
--from <string>	oc-mirror の実行によって生成されたイメージセットアーカイブへのパスを指定して、ターゲットレジストリーにロードします。
-h、--help	ヘルプを表示します。
--ignore-history	イメージをダウンロードしてレイヤーをパックするときに、過去のミラーリングを無視します。増分ミラーリングを無効にし、より多くのデータをダウンロードする可能性があります。
--manifests-only	**ImageContentSourcePolicy** オブジェクトのマニフェストを生成して、ミラーレジストリーを使用するようにクラスターを設定しますが、実際にはイメージをミラーリングしません。このフラグを使用するには、**--from** フラグでイメージセットアーカイブを渡す必要があります。
--max-nested-paths <int>	ネストされたパスを制限する宛先レジストリーのネストされたパスの最大数を指定します。デフォルトは **0** です。
--max-per-registry <int>	レジストリーごとに許可される同時要求の数を指定します。デフォルト値は **6** です。
--oci-insecure-signature-policy	(**--include-local-oci-catalogs** を使用して) ローカル OCI カタログをミラーリングする場合は、署名をプッシュしないでください。
--oci-registries-config	**(--include-local-oci-catalogs** を使用して) ローカル OCI カタログをミラーリングするときに、コピー元の代替レジストリーの場所を指定するためのレジストリー設定ファイルを提供します。
--skip-cleanup	アーティファクトディレクトリーの削除を省略します。
--skip-image-pin	Operator カタログのイメージタグをダイジェストピンに置き換えないでください。
--skip-metadata-check	イメージセットの公開時にメタデータをスキップします。これは、イメージセットが **--ignore-history** で作成された場合にのみ推奨されます。

フラグ	説明
--skip-missing	イメージが見つからない場合は、エラーを報告して実行を中止する代わりにスキップします。イメージセット設定で明示的に指定されたカスタムイメージには適用されません。
--skip-pruning	ターゲットミラーレジストリーからのイメージの自動プルーニングを無効にします。
--skip-verification	ダイジェストの検証を省略します。
--source-skip-tls	ソースレジストリーの TLS 検証を無効にします。
--source-use-http	ソースレジストリーにはプレーン HTTP を使用します。
-v, --verbose <int>	ログレベルの詳細度の数値を指定します。有効な値は **0** - **9** です。デフォルトは **0** です。

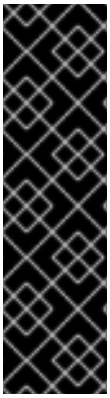
4.4.15. 関連情報

- 非接続環境でのクラスターの更新について

4.5. OC-MIRROR プラグインを使用した非接続インストールのイメージのミラーリング

プライベートレジストリー内の OpenShift Container Platform コンテナーイメージのミラーリングされたセットからクラスターをインストールする場合は、インターネット接続なしで制限されたネットワークでクラスターを稼働させることができます。このレジストリーは、クラスターが実行されているたびに実行されている必要があります。

oc-mirror OpenShift CLI (oc)プラグインを使用できるのと同様に、**oc**-mirror プラグイン v2 を使用して、完全なまたは部分的な非接続環境でイメージをミラーレジストリーにミラーリングすることもできます。公式の Red Hat レジストリーから必要なイメージをダウンロードするには、インターネット接続のあるシステムから oc-mirror を実行する必要があります。



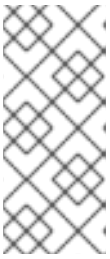
重要

oc-mirror プラグイン v2 はテクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

4.5.1. 前提条件

- Red Hat Quay など、OpenShift Container Platform クラスターをホストする場所に Docker v2-2 をサポートするコンテナーイメージレジストリーを持っている。



注記

Red Hat Quay を使用する場合は、oc-mirror プラグインでバージョン 3.6 以降を使用してください。OpenShift Container Platform (Red Hat Quay ドキュメント) への Red Hat Quay Operator のデプロイ についてのドキュメント を参照してください。レジストリーの選択とインストールについてさらにサポートが必要な場合は、営業担当者または Red Hat サポートにお問い合わせください。

- コンテナーイメージレジストリーの既存のソリューションがない場合、OpenShift Container Platform サブスクライバーは Red Hat Openshift 導入用のミラーレジストリー を受け取ります。このミラーレジストリーはサブスクリプションに含まれており、小規模なコンテナーレジストリーとして機能します。このレジストリーを使用して、非接続インストール用に OpenShift Container Platform の必要なコンテナーイメージをミラーリングできます。

- プロビジョニングされたクラスター内のすべてのマシンはミラーレジストリーにアクセスできる必要があります。レジストリーに到達できない場合、インストール、更新、またはワークロードの再配置などのルーチン操作が失敗する可能性があります。ミラーレジストリーは高可用性の方法で動作し、可用性が OpenShift Container Platform クラスターの実稼働環境の可用性と一致するようにする必要があります。

ワークフローの概要

次の手順は、oc-mirror プラグインを使用してイメージをミラーレジストリーにミラーリングする方法の概要を示しています。

1. イメージセット設定ファイルを作成します。
2. 次のいずれかの方法を使用して、イメージセットをターゲットミラーレジストリーにミラーリングします。
    - イメージセットをターゲットミラーレジストリー（ミラーリングする）に直接ミラーリングします。
        - ディスクに設定されたイメージセットをミラーリング(Mirror-to-Disk)し、**tar** ファイルをターゲット環境に転送してから、イメージセットをターゲットミラーレジストリー(Disk-to-Mirror)にミラーリングします。
3. oc-mirror プラグインが生成したリソースを使用するようにクラスターを設定します。
4. 必要に応じてこれらの手順を繰り返して、ターゲットミラーレジストリーを更新します。

4.5.2. oc-mirror プラグイン v2 について

oc-mirror OpenShift CLI (**oc**)プラグインは、必要なすべての OpenShift Container Platform コンテンツおよびその他のイメージをミラーレジストリーにミラーリングする単一のツールです。

新しいテクノロジープレビューバージョンの oc-mirror を使用するには、**--v2** フラグを oc-mirror プラグイン v2 コマンドラインに追加します。

oc-mirror プラグイン v2 には次の機能があります。

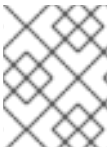
- イメージセット設定で指定された完全なイメージセットが、イメージが以前にミラーリングされたかどうかに関係なく、ミラーリングされたレジストリーにミラーリングされていることを確認します。

- metadata の代わりにキャッシュシステムを使用します。

- 新しいイメージのみをアーカイブに組み込むことで、最小アーカイブサイズを維持します。
- ミラーリング日が選択されたコンテンツでミラーリングアーカイブを生成します。
- 増分の変更だけでなく、完全なイメージセットには **ImageContentSourcePolicy** (ICSP)ではなく、**ImageDigestMirrorSet** (IDMS)、**ImageTagMirrorSet** (ITMS)を生成できます。
- フィルター Operator バージョンをバンドル名で保存します。
- 自動プルーニングを実行しません。V2 には **削除** 機能が追加され、ユーザーにイメージの削除をより詳細に制御できるようになりました。
- **registries.conf** のサポートを紹介します。この変更により、同じキャッシュの使用中に複数の出発者へのミラーリングが容易になります。

4.5.2.1. oc-mirror プラグイン v2 の互換性とサポート

oc-mirror プラグイン v2 は OpenShift Container Platform でサポートされます。



注記

aarch64、**ppc64le**、および **s390x** アーキテクチャーでは、oc-mirror プラグインは OpenShift Container Platform バージョン 4.14 以降でのみサポートされます。

ミラーリングする必要がある OpenShift Container Platform のバージョンに関係なく、使用可能な最新バージョンの oc-mirror プラグインを使用してください。

4.5.3. ミラーホストの準備

イメージミラーリングに oc-mirror プラグイン v2 を使用するには、プラグインをインストールし、コンテナーイメージの認証情報を使用してファイルを作成し、Red Hat からミラーにミラーリングできるようにする必要があります。

4.5.3.1. oc-mirror OpenShift CLI プラグインのインストール

oc-mirror OpenShift CLI プラグインをインストールして、切断された環境でイメージセットを管理します。

前提条件

- OpenShift CLI (**oc**) がインストールされている。完全な非接続環境でイメージセットをミラーリングする場合は、次の点を確認してください。
  - インターネットにアクセスできるホストに oc-mirror プラグインをインストールした。
  - 非接続環境のホストが、ターゲットミラーレジストリーにアクセスできる。
- oc-mirror を使用するオペレーティングシステムで、**umask** パラメーターを **0022** に設定した。
- 使用している RHEL バージョンに適したバイナリーをインストールした。

手順

1. oc-mirror CLI プラグインをダウンロードします。

a. OpenShift Cluster Manager の ダウンロード ページに移動します。

b. **OpenShift 切断インストールツール** セクションで、**OpenShift Client (oc) ミラープラグイン** の **ダウンロード** をクリックしてファイルを保存します。

2. アーカイブを抽出します。

```
$ tar xvzf oc-mirror.tar.gz
```

3. 必要に応じて、プラグインファイルを更新して実行可能にします。

```
$ chmod +x oc-mirror
```

注記

oc-mirror ファイルの名前を変更しないでください。

4. ファイルを **PATH** に配置して、oc-mirror CLI プラグインをインストールします (例: **/usr/local/bin**):。

```
$ sudo mv oc-mirror /usr/local/bin/.
```

検証

- 次のコマンドを実行して、oc-mirror v2 のプラグインが正常にインストールされたことを確認します。

```
$ oc mirror --v2 --help
```

4.5.3.2. イメージのミラーリングを可能にする認証情報の設定

Red Hat からミラーにイメージをミラーリングできるコンテナーイメージレジストリー認証情報ファイルを作成します。

警告

クラスターのインストール時に、このイメージレジストリー認証情報ファイルをプルシークレットとして使用しないでください。クラスターのインストール時にこのファイルを指定すると、クラスター内のすべてのマシンにミラーレジストリーへの書き込みアクセスが付与されます。

警告

このプロセスでは、ミラーレジストリーのコンテナーイメージレジストリーへの書き込みアクセスがあり、認証情報をレジストリープルシークレットに追加する必要があります。

前提条件

- 非接続環境で使用するミラーレジストリーを設定しました。
- イメージをミラーリングするミラーレジストリー上のイメージリポジトリーの場所を特定している。
- イメージのイメージリポジトリーへのアップロードを許可するミラーレジストリーアカウントをプロビジョニングしている。

手順

インストールホストで以下の手順を実行します。

1. **registry.redhat.io** プルシークレットを Red Hat OpenShift Cluster Manager からダウンロードします。
2. JSON 形式でプルシークレットのコピーを作成します。

```
$ cat ./pull-secret | jq . > <path>/<pull_secret_file_in_json> 1
```

1 プルシークレットを保存するフォルダーへのパスおよび作成する JSON ファイルの名前を指定します。

ファイルの内容は以下の例のようになります。

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
```

```
    }
  }
}
```

3. ファイルを **$XDG_RUNTIME_DIR/containers/auth.json** として保存します。

4. ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードまたはトークンを生成します。

```
$ echo -n '<user_name>:<password>' | base64 -w0 (1)
BGVtbYk3ZHAtqXs=
```

(1) **<user_name>** および **<password>** については、レジストリーに設定したユーザー名およびパスワードを指定します。

5. JSON ファイルを編集し、レジストリーについて記述するセクションをこれに追加します。

```
"auths": {
  "<mirror_registry>": { (1)
    "auth": "<credentials>", (2)
    "email": "you@example.com"
  }
},
```

(1) ミラーレジストリーがコンテンツを提供するために使用するレジストリーのドメイン名およびポート (オプション) を指定します。例: **registry.example.com** または **registry.example.com:8443**

(2) ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

ファイルは以下の例のようになります。

```
{
  "auths": {
    "registry.example.com": {
      "auth": "BGVtbYk3ZHAtqXs=",
      "email": "you@example.com"
    },
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
```

```
      "email": "you@example.com"
    }
  }
}
```

4.5.4. イメージセットをミラーレジストリーにミラーリングする

イメージセットをミラーレジストリーにミラーリングすると、必要なイメージが安全で制御された環境で使用可能になり、スムーズなデプロイメント、更新、およびメンテナンスタスクが容易になります。

4.5.4.1. イメージセット設定の構築

oc-mirror プラグイン v2 は、イメージセット設定を入力ファイルとして使用し、ミラーリングに必要なイメージを判別します。

ImageSetConfiguration 入力ファイルの例

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v2alpha1
mirror:
  platform:
    channels:
    - name: stable-4.13
      minVersion: 4.13.10
      maxVersion: 4.13.10
    graph: true
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15
      packages:
       - name: aws-load-balancer-operator
       - name: 3scale-operator
       - name: node-observability-operator
  additionalImages:
   - name: registry.redhat.io/ubi8/ubi:latest
   - name:
registry.redhat.io/ubi9/ubi@sha256:20f695d2a91352d4eaa25107535126727b5945bff38ed36a3e59590f495046f0
```

4.5.4.2. 部分的な非接続環境でのイメージセットのミラーリング

インターネットが制限された環境で oc-mirror プラグイン v2 を使用して、イメージセットをレジストリーにミラーリングできます。

前提条件

- oc-mirror プラグイン v2 を実行している環境でインターネットとミラーレジストリーにアクセスできる。

手順

- 次のコマンドを実行して、指定されたイメージセット設定から指定されたレジストリーにイメージをミラーリングします。

```
$ oc mirror -c isc.yaml --workspace file://<file_path> docker://<mirror_registry_url> --v2 1
```

■

1 イメージの保存先および削除元となるミラーレジストリーの URL またはアドレスを指定します。

検証

1. < **file_path> ディレクトリーに生成された working-dir ディレクトリー内の cluster-resources ディレクトリーに移動し** ます。
2. **ImageDigestMirrorSet**、**ImageTagMirrorSet**、および **CatalogSource** リソースに YAML ファイルが存在することを確認します。

次のステップ

- oc-mirror プラグインが生成したリソースを使用するようにクラスターを設定します。

4.5.4.3. 完全な非接続環境でのイメージセットのミラーリング

OpenShift Container Platform クラスターがパブリックインターネットにアクセスできない完全な非接続環境でイメージセットをミラーリングできます。

1. **Mirror to disk**: ミラーリング用のイメージセットを含むアーカイブを準備します。インターネットアクセスは必要ありません。
2. **手動手順**: アーカイブを切断されたミラーレジストリーのネットワークに移動します。
3. **ディスクとミラーリング**: アーカイブからターゲットの非接続レジストリーにミラーリングするには、ミラーレジストリーにアクセスできる環境から oc-mirror プラグイン v2 を実行します。

4.5.4.3.1. ミラーからディスクへのミラーリング

oc-mirror プラグインを使用して、イメージセットを生成し、コンテンツをディスクに保存できます。生成されたイメージセットは、非接続環境に転送され、ターゲットレジストリーにミラーリングされます。

oc-mirror プラグイン v2 は、イメージセット設定で指定されたソースからコンテナーイメージを取得し、ローカルディレクトリーの tar アーカイブにパックします。

手順

- 次のコマンドを実行して、指定されたイメージセット設定からディスクにイメージをミラーリングします。

```
$ oc mirror -c isc.yaml file://<file_path> --v2 1
```

1 必要なファイルパスを追加します。

検証

1. 生成された &lt **;file_path&gt**; ディレクトリーに移動します。
2. アーカイブファイルが生成されていることを確認します。

次のステップ

- oc-mirror プラグインが生成したリソースを使用するようにクラスターを設定します。

4.5.4.3.2. ディスクからミラーへのミラーリング

oc-mirror プラグイン v2 を使用して、イメージセットをディスクからターゲットミラーレジストリーにミラーリングできます。

oc-mirror プラグイン v2 はローカルディスクからコンテナーイメージを取得して、指定されたミラーレジストリーに転送します。

手順

- 次のコマンドを実行して、ディスク上のイメージセットファイルを処理し、コンテンツをターゲットミラーレジストリーにミラーリングします。

```
$ oc mirror -c isc.yaml --from file://<file_path> docker://<mirror_registry_url> --v2 1
```

1 イメージの保存先および削除元となるミラーレジストリーの URL またはアドレスを指定します。

検証

1. < **file_path> ディレクトリーに生成された working-dir ディレクトリー内の cluster-resources ディレクトリーに移動し** ます。
2. **ImageDigestMirrorSet**、**ImageTagMirrorSet**、および **CatalogSource** リソースに YAML ファイルが存在することを確認します。

次のステップ

- oc-mirror プラグインが生成したリソースを使用するようにクラスターを設定します。

4.5.5. 関連情報

- OpenShift Update Service を使用した非接続環境でのクラスターの更新

4.5.6. v2 によって生成されるカスタムリソースについて

oc-mirror プラグイン v2 では、タグが参照する少なくとも 1 つのイメージが見つかると、**ImageDigestMirrorSet** (IDMS)と **ImageTagMirrorSet** (ITMS)がデフォルトで生成されます。これらのセットには、リリースのダイジェストまたはタグで参照されるイメージのミラー、Operator カタログ、および追加のイメージが含まれます。

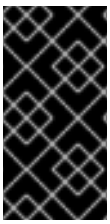
ImageDigestMirrorSet (IDMS)はミラーレジストリーをソースレジストリーにリンクし、ダイジェスト仕様を使用してイメージプルシークレットを転送します。ただし、**ImagetagMirrorSet** (ITMS)リソースは、イメージタグを使用してイメージのプル要求をリダイレクトします。

CatalogSource リソースは、Operator Lifecycle Manager (OLM) によって使用され、ミラーレジストリーで使用可能な Operator に関する情報を取得します。

OSUS サービスは、**UpdateService** リソースを使用して、切断された環境に Cincinnati グラフを提供します。

4.5.6.1. oc-mirror が生成したリソースを使用するためのクラスター設定

イメージセットをミラーレジストリーにミラーリングした後、生成された **ImageDigestMirrorSet** (IDMS)、**ImageTagMirrorSet** (ITMS)、**CatalogSource**、および **UpdateService** をクラスターに適用する必要があります。



重要

oc-mirror プラグイン v2 では、IDMS および ITMS ファイルは、oc-mirror プラグイン v1 の ICSP ファイルとは異なり、イメージセット全体をカバーします。したがって、IDMS および ITMS ファイルには、増分ミラーリング時に新しいイメージのみを追加する場合でも、セットのすべてのイメージが含まれます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- 以下のコマンドを実行して、results ディレクトリーからクラスターに YAML ファイルを適用します。

```
$ oc apply -f <path_to_oc-mirror_workspace>/working-dir/cluster-resources
```

検証

1. 次のコマンドを実行して、**ImageDigestMirrorSet** リソースが正常にインストールされていることを確認します。

```
$ oc get imagedigestmirrorset
```

2. 次のコマンドを実行して、**ImageTagMirrorSet** リソースが正常にインストールされていることを確認します。

```
$ oc get imagetagmirrorset
```

3. 以下のコマンドを実行して、**CatalogSource** リソースが正常にインストールされたことを確認します。

```
$ oc get catalogsource -n openshift-marketplace
```

4.5.7. 非接続環境でのイメージの削除

oc-mirror プラグイン v2 を使用する前に、以前にデプロイされたイメージを削除する必要があります。oc-mirror プラグイン v2 は自動プルーニングを実行しなくなります。

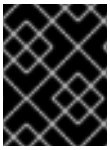
oc-mirror プラグイン v2 を使用する場合は、イメージ設定を削除するには **DeleteImageSetConfiguration** ファイルを作成する必要があります。これにより、**ImageSetConfig.yaml** で変更する際に、必要なイメージやデプロイされたイメージが誤って削除されるのを防ぐことができます。

次の例では、**DeleteImageSetConfiguration** は以下を削除します。

- OpenShift Container Platform リリース 4.13.3 のすべてのイメージ。
- カタログイメージ **redhat-operator-index v4.12**。
- **aws-load-balancer-operator** v0.0.1 バンドルとそのすべての関連イメージ。
- 対応するダイジェストで参照される **ubi** および **ubi-minimal** の追加のイメージ。

例： DeleteImageSetConfig

```
apiVersion: mirror.openshift.io/v2alpha1
kind: DeleteImageSetConfiguration
delete:
  platform:
    channels:
      - name: stable-4.13
        minVersion: 4.13.3
        maxVersion: 4.13.3
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
      packages:
      - name: aws-load-balancer-operator
          minVersion: 0.0.1
          maxVersion: 0.0.1
  additionalImages:
    - name:
registry.redhat.io/ubi8/ubi@sha256:bce7e9f69fb7d4533447232478fd825811c760288f87a35699f9c8f03
0f2c1a6
    - name: registry.redhat.io/ubi8/ubi-
minimal@sha256:8bedbe742f140108897fb3532068e8316900d9814f399d676ac78b46e740e34e
```



重要

ミラーツーディスクおよびディスク間ワークフローを使用してミラーリングの問題を減らすことを検討してください。

イメージ削除ワークフローでは、oc-mirror プラグイン v2 はイメージのマニフェストのみを削除します。これにより、レジストリーで占有されるストレージが削減されません。

マニフェストが削除されたものなど、不要なイメージからストレージ領域を解放するには、コンテナーレジストリーでガベージコレクターを有効にする必要があります。ガベージコレクターを有効にすると、レジストリーはマニフェストへの参照を持たないイメージ Blob を削除するため、削除された BLOB によって過去に占有されたストレージが削減されます。ガベッジコレクターの有効化は、コンテナーレジストリーによって異なります。

4.5.7.1. 非接続環境でのイメージの削除

oc-mirror プラグイン v2 を使用して切断された環境からイメージを削除するには、次の手順に従います。

手順

1. 以前のイメージを削除する YAML ファイルを作成します。

```
$ oc mirror delete --config delete-image-set-config.yaml --workspace
file://<previously_mirrored_work_folder> --v2 --generate docker://<remote_registry>
```

ここでは、以下のようになります。

- **<previously_mirrored_work_folder** >: ミラーリングプロセス中にイメージがミラーリングまたは保存されていたディレクトリーを使用します。
- **<remote_registry** >: イメージを削除するリモートコンテナーレジストリーの URL またはアドレスを挿入します。

2. 作成された **<previously_mirrored_work_folder>/delete ディレクトリー** に移動します。
3. **delete-images.yaml** ファイルが生成されていることを確認します。
4. ファイルにリストされている各イメージがクラスターで不要になり、レジストリーから安全に削除できることを確認します。
5. **削除** YAML ファイルを生成したら、リモートレジストリーからイメージを削除します。

```
$ oc mirror delete --v2 --delete-yaml-file <previously_mirrored_work_folder>/delete/delete-
images.yaml docker:/ <remote_registry>
```

ここでは、以下のようになります。

- & **lt;previously_mirrored_work_folder>: 以前** にミラーリングされたワークフォルダーを指定します。

重要

ミラー間の手順を使用する場合、イメージはローカルにキャッシュされないため、ローカルキャッシュからイメージを削除することはできません。

4.5.8. ミラーリング用に選択したイメージを確認する

oc-mirror プラグイン v2 を使用して、実際にイメージをミラーリングしないテストラン（ドライラン）を実行できます。これにより、ミラーリングされるイメージの一覧を確認できます。ドライランを使用して、イメージセット設定のエラーを早期にキャッチすることもできます。ミラーツーディスクのワークフローでドライランを実行する場合、oc-mirror プラグイン v2 は、イメージセット内のすべてのイメージがキャッシュで利用可能かどうかを確認します。欠落しているイメージは **missing.txt** ファイルに一覧表示されます。ミラーリング前にドライランが実行されると、**missing.txt** ファイルと **mapping.txt** ファイルの両方に同じイメージのリストが含まれます。

4.5.8.1. oc-mirror プラグイン v2 のドライランの実行

イメージをミラーリングせずにドライランを実行して、イメージセット設定を確認します。これにより、設定が正しく、意図しない変更を防ぐことができます。

手順

- テスト実行を実行するには、**oc mirror** コマンドを実行し、コマンドに **--dry-run** 引数を追加します。

```
$ oc mirror -c <image_set_config_yaml> --from file://<oc_mirror_workspace_path>
docker://<mirror_registry_url> --dry-run --v2
```

ここでは、以下のようになります。

- **<image_set_config_yaml&** gt;: 作成したイメージセット設定ファイルを使用します。
- **<oc_mirror_workspace_path&gt**;: ワークスペースパスのアドレスを挿入します。
- **<mirror_registry_url** >: イメージを削除するリモートコンテナーレジストリーの URL またはアドレスを挿入します。

出力例

```
$ oc mirror --config /tmp/isc_dryrun.yaml file://<oc_mirror_workspace_path> --dry-run --v2

[INFO]   : :warning:  --v2 flag identified, flow redirected to the oc-mirror v2 version. This is Tech Preview, it is still under development and it is not production ready.
[INFO]   : :wave: Hello, welcome to oc-mirror
[INFO]   : :gear:  setting up the environment for you...
[INFO]   : :twisted_rightwards_arrows: workflow mode: mirrorToDisk
[INFO]   : :sleuth_or_spy:  going to discover the necessary images...
[INFO]   : :mag: collecting release images...
[INFO]   : :mag: collecting operator images...
[INFO]   : :mag: collecting additional images...
[WARN]   : :warning:  54/54 images necessary for mirroring are not available in the cache.
[WARN]   : List of missing images in : CLID-19/working-dir/dry-run/missing.txt.
please re-run the mirror to disk process
[INFO]   : :page_facing_up: list of all images for mirroring in : CLID-19/working-dir/dry-run/mapping.txt
[INFO]   : mirror time     : 9.641091076s
[INFO]   : :wave: Goodbye, thank you for using oc-mirror
```

検証

1. 生成されたワークスペースディレクトリーに移動します。

```
$ cd <oc_mirror_workspace_path>
```

2. 生成された **mapping.txt** および **missing.txt** ファイルを確認します。これらのファイルには、ミラーリングされるすべてのイメージのリストが含まれています。

4.5.9. サポートを依頼する利点

Enclave サポートは、ネットワークの特定の部分への内部アクセスを制限します。ファイアウォール境界を介したインバウンドおよびアウトバウンドトラフィックアクセスを可能にする demilitarized zone (DMZ)ネットワークとは異なり、ファイアウォールの境界を超えません。

重要

Devfile のサポートはテクノロジープレビュー機能でのみ利用可能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

新しいエンクターサポート機能は、少なくとも 1 つの中間切断されたネットワークの背後でセキュリティーが保護された複数のエンクリーブに対してミラーリングが必要なシナリオです。

Enclave サポートには、以下の利点があります。

- 複数のエンクリーブのコンテンツをミラーリングし、単一の内部レジストリーでこれを一元化できます。一部のお客様はミラーリングされたコンテンツに対してセキュリティーチェックを実行したいので、このセットアップでは、これらのチェックを一度に実行できます。コンテンツは、ダウンストリームのエンクトにミラーリングされる前にレポートされます。
- 一元化された内部レジストリーからコンテンツを直接ミラーリングし、エンクリーブごとにインターネットからミラーリングプロセスを再起動せずに拡張できます。
- ネットワークステージ間のデータ転送を最小限に抑えて、BLOB またはイメージがあるステージから別のステージに 1 回だけ転送されるようにすることができます。

4.5.9.1. エンクリーブへのミラーリング

エンクリーブにミラーリングするときは、まず必要なイメージを 1 つ以上のサーバーからエンタープライズ中央レジストリーに転送する必要があります。

中央レジストリーは、安全なネットワーク、特に非接続環境内であり、パブリックインターネットに直接リンクされていません。ただし、ユーザーはパブリックインターネットにアクセスできる環境で **oc mirror** を実行する必要があります。

手順

1. 非接続環境で oc-mirror プラグイン v2 を実行する前に、**registries.conf** ファイルを作成します。ファイルの TOML 形式を以下の仕様に示します。

注記

ファイルは **$HOME/.config/containers/registries.conf** または **/etc/containers/registries.conf** の下に保存することを推奨します。

registries.confの例

```
[[registry]]
location="registry.redhat.io"
[[registry.mirror]]
location="<enterprise-registry.in>"

[[registry]]
```

```
location="quay.io"
[[registry.mirror]]
location="<enterprise-registry.in>"
```

2. ミラーアーカイブを生成します。

   a. すべての OpenShift Container Platform コンテンツを < **file_path>/enterprise-content** 下のディスク上のアーカイブに収集するには、以下のコマンドを実行します。

   ```
   $ oc mirror --v2 -c isc.yaml file://<file_path>/enterprise-content
   ```

   isc.yaml の例：

   ```
   apiVersion: mirror.openshift.io/v2alpha1
   kind: ImageSetConfiguration
   mirror:
     platform:
       architectures:
         - "amd64"
       channels:
         - name: stable-4.15
           minVersion: 4.15.0
           maxVersion: 4.15.3
   ```

   アーカイブが生成されたら、非接続環境に転送されます。トランスポートメカニズムは oc-mirror プラグイン v2 の一部ではありません。エンタープライズネットワークの管理者は、転送戦略を決定します。

   場合によっては、ディスクがある場所から物理的に接続解除され、非接続環境で別のコンピューターに接続されるという点で、手動で転送が行われます。その他の場合は、SFTP (Secure File Transfer Protocol)またはその他のプロトコルが使用されます。

3. アーカイブの転送が完了した後、次の例に示すように、関連するアーカイブコンテンツをレジストリーにミラーリングするために、oc-mirror プラグイン v2 を再度実行できます（例では**entrerpise_registry.in** ）。

   ```
   $ oc mirror --v2 -c isc.yaml --from file://<disconnected_environment_file_path>/enterprise-content docker://<enterprise_registry.in>/
   ```

   ここでは、以下のようになります。

   - **--from** は、アーカイブを含むフォルダーを指します。file:// で始まり **ます**。
   - **docker:**// はミラーリングの宛先であり、最終的な引数になります。これは docker レジストリーであるためです。
   - **-c** (**--config**)は必須の引数です。これにより、oc-mirror プラグイン v2 は最終的にアーカイブのサブ部分のみをレジストリーにミラーリングできます。1 つのアーカイブに複数の OpenShift Container Platform リリースが含まれる可能性がありますが、非接続環境またはエンプトされるのはごくわずかにミラーリングする可能性があります。

4. enclave にミラーリングするコンテンツを記述する **imageSetConfig** YAML ファイルを準備します。

   isc-enclave.yaml の例

```
apiVersion: mirror.openshift.io/v2alpha1
kind: ImageSetConfiguration
mirror:
  platform:
    architectures:
      - "amd64"
    channels:
      - name: stable-4.15
        minVersion: 4.15.2
        maxVersion: 4.15.2
```

切断されたレジストリーにアクセスできるマシンで、oc-mirror プラグイン v2 を実行する必要があります。上記の例では、非接続環境 **enterprise-registry.in** にアクセスできます。

5. グラフ URL の更新
graph:true を使用している場合、oc-mirror プラグイン v2 は **cincinnati** API エンドポイントに到達しようとします。この環境は切断されているため、環境変数 **UPDATE_URL_OVERRIDE** をエクスポートして、OpenShift Update Service (OSUS)の URL を参照してください。

```
$ export UPDATE_URL_OVERRIDE=https://<osus.enterprise.in>/graph
```

OpenShift クラスターで OSUS を設定する方法は、OpenShift Update Service を使用した非接続環境でのクラスターの更新 を参照してください。

6. enclave のエンタープライズレジストリーからミラーアーカイブを生成します。
enclave1 のアーカイブを準備するために、ユーザーはその enclave 固有の **imageSetConfiguration** を使用して、エンタープライズ切断された環境で oc-mirror プラグイン v2 を実行します。これにより、その enclave が必要とするイメージのみがミラーリングされます。

```
$ oc mirror --v2 -c isc-enclave.yaml
file:///disk-enc1/
```

このアクションでは、すべての OpenShift Container Platform コンテンツをアーカイブに収集し、ディスクにアーカイブを生成します。

7. アーカイブが生成されたら、**enclave1** ネットワークに転送されます。トランスポートメカニズムは oc-mirror プラグイン v2 の責任ではありません。

8. エンクリーブレジストリーにコンテンツをミラーリングします。
アーカイブの転送が完了した後、ユーザーは oc-mirror プラグイン v2 を再度実行して、関連するアーカイブコンテンツをレジストリーにミラーリングできます。

```
$ oc mirror --v2 -c isc-enclave.yaml --from file://local-disk docker://registry.enc1.in
```

enclave1 の OpenShift Container Platform クラスターの管理者は、そのクラスターをインストールまたはアップグレードする準備が整いました。

4.5.10. Operator カタログでのフィルタリングがどのように機能するか

oc-mirror プラグイン v2 は、**imageSetConfig** の情報を処理してミラーリングするバンドルの一覧を選択します。

oc-mirror プラグイン v2 がミラーリングのバンドルを選択する場合、そのバンドルはグループバージョ

ンの Kind (GVK)またはバンドルの依存関係ではなく、ミラーリングセットから省略されます。代わりに、ユーザーの指示を厳密に準拠させます。必要な依存パッケージとそのバージョンを明示的に指定する必要があります。

バンドルバージョンは、通常、セマンティックバージョニング標準(SemVer)を使用し、チャネル内のバンドルをバージョンごとに並べ替えることができます。**ImageSetConfig** の特定の範囲内にある外国を選択できます。

この選択アルゴリズムにより、oc-mirror プラグイン v1 と比較して一貫した結果が保証されます。ただし、**置換**、スキップ、および **skip Range** などのアップグレードグラフの詳細は含まれません。この方法は OLM アルゴリズムとは異なります。**minVersion** と **maxVersion** の間でアップグレードパスが短くなる可能性があるため、クラスターのアップグレードに必要な以上のバンドルをミラーリングする可能性があります。

表4.4 以下の表を使用して、さまざまなシナリオに含まれるバンドルバージョンを確認します。

ImageSetConfig Operator のフィルタリング	予想されるバンドルバージョン
シナリオ 1 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10	カタログの各パッケージについて、そのパッケージのデフォルトチャネルのヘッドバージョンに対応する 1 バンドルです。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 full: true	指定されたカタログのすべてのチャネルのすべてのバンドル
シナリオ 3 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 packages: - name: compliance-operator	そのパッケージのデフォルトチャネルのヘッドバージョンに対応する 1 つのバンドル

ImageSetConfig Operator のフィルタリング	予想されるバンドルバージョン
シナリオ 4 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 full: true - packages: - name: elasticsearch-operator	指定されたパッケージに対するすべてのチャネルのすべてのバンドル
シナリオ 5 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator minVersion: 5.6.0	デフォルトチャネル内のすべてのバンドル(**minVersion** から、グラフのアップグレードから最短のパスに依存しないパッケージのチャネルヘッドまで)。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator maxVersion: 6.0.0	対象のパッケージの **maxVersion** よりも小さいデフォルトチャネル内のすべてのバンドル。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator minVersion: 5.6.0 maxVersion: 6.0.0	そのパッケージの **minVersion** と **maxVersion** の間に、デフォルトチャネル内のすべてのバンドル。複数のチャネルがフィルターに含まれている場合でも、チャネルのヘッドは含まれません。

ImageSetConfig Operator のフィルタリング	予想されるバンドルバージョン
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable	そのパッケージの選択したチャネルのヘッドバンドル。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 full: true - packages: - name: elasticsearch-operator channels: - name: 'stable-v0'	指定したパッケージおよびチャネルのすべてのバンドル。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable - name: stable-5.5	そのパッケージの選択した各チャネルのヘッドバンドル。

ImageSetConfig Operator のフィルタリング	予想されるバンドルバージョン
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0	そのパッケージの選択したチャネル内で、**minVersion** で始まるすべてのバージョンからチャネルヘッドまで。このシナリオは、アップグレードグラフからの最短パスに依存しません。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable maxVersion: 6.0.0	そのパッケージの選択したチャネル内では、**maxVersion** までのすべてのバージョン（アップグレードグラフからの最短パスに依存しない）。複数のチャネルがフィルターに含まれている場合でも、チャネルのヘッドは含まれません。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	そのパッケージの選択したチャネル内では、アップグレードグラフからの最短パスではなく、**minVersion** と **maxVersion** の間にあるすべてのバージョン。複数のチャネルがフィルターに含まれている場合でも、チャネルのヘッドは含まれません。

ImageSetConfig Operator のフィルタリング	予想されるバンドルバージョン
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14 packages: - name: aws-load-balancer-operator bundles: - name: aws-load-balancer-operator.v1.1.0 - name: 3scale-operator bundles: - name: 3scale-operator.v0.10.0-mas	各パッケージに指定されたバンドルのみがフィルターに含まれます。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	このシナリオは使用しないでください。チャネルによるフィルタリングや、**minVersion** または **maxVersion** を使用したパッケージによるフィルタリングは許可されません。
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	このシナリオは使用しないでください。**full:true** および **minVersion** または **maxVersion** を使用してフィルタリングすることはできません。

ImageSetConfig Operator のフィルタリング	予想されるバンドルバージョン
シナリオ 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 full: true packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	このシナリオは使用しないでください。**full:true** および **minVersion** または **maxVersion** を使用してフィルタリングすることはできません。

4.5.11. oc-mirror プラグイン v2 の イメージセット 設定パラメーター

oc-mirror プラグインには、ミラーリングするイメージを定義するイメージセット設定ファイルが必要です。次の表に、**ImageSetConfiguration** リソースで使用可能なパラメーターを示します。

注記

minVersion および **maxVersion** プロパティーを使用して特定の Operator バージョン範囲をフィルターすると、複数のチャネルヘッドエラーが発生する可能性があります。エラーメッセージには、**multiple channel heads** があることが示されます。これは、フィルターを適用すると、Operator の更新グラフが切り捨てられるためです。

すべての Operator チャネルに、1 つのエンドポイント (つまり最新バージョンの Operator) を持つ更新グラフを構成するバージョンが含まれている必要があります。これは Operator Lifecycle Manager によって要求される要件です。フィルター範囲を適用すると、更新グラフが 2 つ以上の個別のグラフ、または複数のエンドポイントを持つグラフに変換されることがあります。

このエラーを回避するには、最新バージョンの Operator を除外しないでください。それでもエラーが発生する場合は、Operator に応じて、**maxVersion** プロパティーを増やすか、**minVersion** プロパティーを減らす必要があります。Operator グラフはそれぞれ異なる可能性があるため、エラーが解決するまでこれらの値を調整する必要がある場合があります。

表4.5 ImageSetConfiguration パラメーター

パラメーター	説明	値
apiVersion	**ImageSetConfiguration** コンテンツの API バージョン。	文字列の例：**mirror.openshift.io/v2alpha1**

パラメーター	説明	値
archiveSize	イメージセット内の各アーカイブファイルの最大サイズ (GiB 単位)。	整数例： **4**
mirror	イメージセットの設定。	オブジェクト
mirror.additionalImages	イメージセットの追加のイメージ設定。	オブジェクトの配列 以下に例を示します。 `additionalImages:` `  - name: registry.redhat.io/ubi8/ubi:latest`
mirror.additionalImages.name	ミラーリングするイメージのタグまたはダイジェスト。	例: **registry.redhat.io/ubi8/ubi:latest**
mirror.blockedImages	ミラーリングからブロックするイメージの完全なタグ、ダイジェスト、またはパターン。	文字列の配列： **docker.io/library/alpine**
mirror.operators	イメージセットの Operators 設定。	オブジェクトの配列 以下に例を示します。 `operators:` `  - catalog: registry.redhat.io/redhat/redhat-operator-index:4.16` `    packages:` `      - name: elasticsearch-operator` `minVersion: '2.4.0'`

パラメーター	説明	値
mirror.operators.catalog	イメージセットに含める Operator カタログ。	たとえば、**registry.redhat.io/redhat/redhat-operator-index:v4.15** です。
mirror.operators.full	**true** の場合、完全なカタログ、Operator パッケージ、または Operator チャネルをダウンロードします。	boolean デフォルト値は **false** です。
mirror.operators.packages	Operator パッケージ設定	オブジェクトの配列 以下に例を示します。 operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:4.16 packages: - name: elasticsearch-operator minVersion: '5.2.3-31'
mirror.operators.packages.name	イメージセットに含める Operator パッケージ名	文字列の例：**elasticsearch-operator**
mirror.operators.packages.channels	Operator パッケージのチャネル設定。	Object
mirror.operators.packages.channels.name	イメージセットに含める、パッケージ内で一意の Operator チャネル名。	String Eample: **fast** または **stable-v4.15**
mirror.operators.packages.channels.maxVersion	Operator が存在するすべてのチャネルでミラーリングする最上位バージョンの Operator。	文字列の例：**5.2.3-31**
mirror.operators.packages.channels.minVersion	存在するすべてのチャネル間でミラーリングする Operator の最低バージョン。	文字列の例：**5.2.3-31**

パラメーター	説明	値
mirror.operators.packages.maxVersion	Operator が存在するすべてのチャネルでミラーリングする最上位バージョンの Operator。	文字列の例：**5.2.3-31**
mirror.operators.packages.minVersion	存在するすべてのチャネル間でミラーリングする Operator の最低バージョン。	文字列の例：**5.2.3-31**
mirror.operators.packages.bundles	選択したバンドルの設定	オブジェクトの配列 以下に例を示します。 operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:4.16 packages: - name: 3scale-operator bundles: - name: 3scale-operator.v0.10.0-mas
mirror.operators.packages.bundles.name	ミラー用に選択されたバンドルの名前（カタログに表示される）。	文字列の例：**3scale-operator.v0.10.0-mas**
mirror.operators.targetCatalog	参照されるカタログをミラーリングするための代替名とオプションの namespace 階層。	例: **my-namespace/my-operator-catalog**

パラメーター	説明	値
mirror.operators.targetCatalogSourceTemplate	oc-mirror プラグイン v2 によって生成された catalogSource カスタムリソースを完了するために使用するテンプレートのディスク上のパス。	文字列の例： **/tmp/catalog-source_template.yaml** テンプレートファイルの例： apiVersion: operators.coreos.com/v2alpha1 kind: CatalogSource metadata: name: discarded namespace: openshift-marketplace spec: image: discarded sourceType: grpc updateStrategy: registryPoll: interval: 30m0s
mirror.operators.targetTag	**targetName** または **targetCatalog** に追加する代替タグ。	文字列例： **v1**
mirror.platform	イメージセットのプラットフォーム設定。	オブジェクト

パラメーター	説明	値
mirror.platform.architectures	ミラーリングするプラットフォームリリースペイロードのアーキテクチャー。	文字列の配列の例： architectures: - amd64 - arm64 - multi - ppc64le - s390x デフォルト値は **amd64** です。値 multi を指定すると、使用可能なすべてのアーキテクチャーでミラーリングがサポートされ、個別のアーキテクチャーを指定する必要がなくなります。
mirror.platform.channels	イメージセットのプラットフォームチャネル設定。	オブジェクトの配列の例： channels: - name: stable-4.12 - name: stable-4.16
mirror.platform.channels.full	**true** の場合、**minVersion** をチャネルの最初のリリースに設定し、**maxVersion** をチャネルの最後のリリースに設定します。	boolean デフォルト値は **false**です
mirror.platform.channels.name	リリースチャネルの名前。	文字列の例： **stable-4.15**
mirror.platform.channels.minVersion	ミラーリングされる参照プラットフォームの最小バージョン。	文字列例： **4.12.6**
mirror.platform.channels.maxVersion	ミラーリングされる参照プラットフォームの最上位バージョン。	文字列の例： **4.15.1**
mirror.platform.channels.shortestPath	最短パスミラーリングまたはフルレンジミラーリングを切り替えます。	boolean デフォルト値は **false**です

パラメーター	説明	値
mirror.platform.channels.type	ミラーリングするプラットフォームのタイプ。	文字列の例： **ocp** または **okd**。デフォルトは **ocp** です。
mirror.platform.graph	OSUS グラフがイメージセットに追加され、その後ミラーに公開されるかどうかを示します。	boolean デフォルト値は **false**です

4.5.11.1. イメージセット 設定パラメーターの削除

oc-mirror プラグイン v2 を使用するには、ミラーレジストリーから削除するイメージを定義するイメージセット設定ファイルを削除する必要があります。次の表に、**ImageSetConfiguration** リソースで使用可能なパラメーターを示します。

表4.6 DeleteImageSetConfiguration パラメーター

パラメーター	説明	値
apiVersion	**DeleteImageSetConfiguration** コンテンツの API バージョン。	文字列の例： **mirror.openshift.io/v2alpha1**
delete	削除するイメージセットの設定。	Object
delete.additionalImages	delete イメージセットの追加のイメージ設定。	オブジェクトの配列の例： `additionalImages:` `  - name: registry.redhat.io/ubi8/ubi:latest`
delete.additionalImages.name	削除するイメージのタグまたはダイジェスト。	例: **registry.redhat.io/ubi8/ubi:latest**

パラメーター	説明	値
delete.operators	delete イメージセットの Operator 設定。	オブジェクトの配列の例： operators: - catalog: registry.redhat.io/redhat/redhat-operator-index: {product-version} packages: - name: elasticsearch-operator minVersion: '2.4.0'
delete.operators.catalog	delete イメージセットに含める Operator カタログ。	たとえば、**registry.redhat.io/redhat/redhat-operator-index:v4.15** です。
delete.operators.full	true の場合、完全なカタログ、Operator パッケージ、または Operator チャネルを削除します。	boolean デフォルト値は **false**です
delete.operators.packages	Operator パッケージ設定	オブジェクトの配列の例： operators: - catalog: registry.redhat.io/redhat/redhat-operator-index: {product-version} packages: - name: elasticsearch-operator minVersion: '5.2.3-31'

パラメーター	説明	値
delete.operators.packages.name	削除イメージセットに含める Operator パッケージ名。	文字列の例： **elasticsearch-operator**
delete.operators.packages.channels	Operator パッケージのチャネル設定。	Object
delete.operators.packages.channels.name	イメージセットに含める、パッケージ内で一意の Operator チャネル名。	文字列の例： **fast** または **stable-v4.15**
delete.operators.packages.channels.maxVersion	選択したチャネル内で削除する Operator の最上位バージョン。	文字列の例： **5.2.3-31**
delete.operators.packages.channels.minVersion	削除する Operator の最低バージョン。それが存在する選択内で削除します。	文字列の例： **5.2.3-31**
delete.operators.packages.maxVersion	Operator が存在するすべてのチャネルでミラーリングする最上位バージョンの Operator。	文字列の例： **5.2.3-31**
delete.operators.packages.minVersion	存在するすべてのチャネル間でミラーリングする Operator の最低バージョン。	文字列の例： **5.2.3-31**

パラメーター	説明	値
delete.operators.packages.bundles	選択したバンドルの設定	オブジェクトの配列 同じ Operator に対してチャネルとバンドルの両方を選択することはできません。 以下に例を示します。 operators: - catalog: registry.redhat.io/redhat/redhat-operator-index: {product-version} packages: - name: 3scale-operator bundles: - name: 3scale-operator.v0.10.0-mas
delete.operators.packages.bundles.name	削除するように選択されたバンドルの名前（カタログに表示されるため）	文字列の例： **3scale-operator.v0.10.0-mas**
delete.platform	イメージセットのプラットフォーム設定。	Object
delete.platform.architectures	ミラーリングするプラットフォームリリースペイロードのアーキテクチャー。	文字列の配列の例： architectures: - amd64 - arm64 - multi - ppc64le - s390x デフォルト値は **amd64** です。

パラメーター	説明	値
delete.platform.channels	イメージセットのプラットフォームチャネル設定。	オブジェクトの配列 以下に例を示します。 channels: - name: stable-4.12 - name: stable-4.16
delete.platform.channels.full	**true** の場合、**minVersion** をチャネルの最初のリリースに設定し、**maxVersion** をチャネルの最後のリリースに設定します。	boolean デフォルト値は **false**です
delete.platform.channels.name	リリースチャネルの名前。	文字列の例：**stable-4.15**
delete.platform.channels.minVersion	ミラーリングされる参照プラットフォームの最小バージョン。	文字列例： **4.12.6**
delete.platform.channels.maxVersion	ミラーリングされる参照プラットフォームの最上位バージョン。	文字列の例：**4.15.1**
delete.platform.channels.shortestPath	最短パスの削除とフル範囲の削除を切り替えます。	boolean デフォルト値は **false**です
delete.platform.channels.type	削除するプラットフォームのタイプ	文字列の例： **ocp** または **okd** のデフォルト値は **ocp**です。
delete.platform.graph	OSUS グラフもミラーレジストリーで削除されるかどうかを決定します。	boolean デフォルト値は **false**です

4.5.12. oc-mirror プラグイン v2 のコマンドリファレンス

以下の表は、oc-mirror プラグイン v2 の **oc mirror** サブコマンドとフラグについて説明しています。

表4.7 oc-mirror プラグイン v2 のサブコマンドとフラグ

サブコマンド	説明
help	サブコマンドに関するヘルプを表示します。
version	oc-mirror バージョンを出力します。

サブコマンド	説明
delete	リモートレジストリーおよびローカルキャッシュ内のイメージを削除します。

表4.8 oc mirror フラグ

フラグ	説明
--authfile	認証ファイルの文字列パスを表示します。デフォルトは **${XDG_RUNTIME_DIR}/containers/auth.json** です。
-c, **--config <string>**	イメージセット設定ファイルへのパスを指定します。
--dest-tls-verify	コンテナーレジストリーまたはデーモンにアクセスする際に、HTTPS を必要とし、証明書を検証します。
--dry-run	イメージをミラーリングせずにアクションを出力します
--from <string>	oc-mirror プラグイン v2 を実行してターゲットレジストリーをロードすることにより生成されたイメージセットアーカイブへのパスを指定します。
-h、**--help**	ヘルプの表示
--loglevel	文字列ログレベルを表示します。サポートされる値には、info、debug、trace、error が含まれます。デフォルトは **info** です。
-p、--port	oc-mirror プラグイン v2 ローカルストレージインスタンスによって使用される HTTP ポートを決定します。デフォルトは **0.7** です。
--max-nested-paths <int>	ネストされたパスを制限する宛先レジストリーのネストされたパスの最大数を指定します。デフォルトは **0** です。
--secure-policy	デフォルト値は **false** です。デフォルト以外の値を設定すると、コマンドは署名の検証の安全なポリシーである署名を検証を有効にします。
Since	指定された日付以降のすべての新しいコンテンツが含まれます（形式：**yyyy-mm-dd**）。指定しない場合、以前のミラーリング以降の新しいコンテンツがミラーリングされます。
--src-tls-verify	コンテナーレジストリーまたはデーモンにアクセスする際に、HTTPS を必要とし、証明書を検証します。
--strict-archive	デフォルト値は **false** です。値を設定すると、コマンドは **imageSetConfig** カスタムリソース(CR)で設定された **archiveSize** よりも厳密に小さいアーカイブを生成します。アーカイブしているファイルが **archiveSize** (GB)を超える場合、ミラーリングはエラーに存在します。

フラグ	説明
-V, --version	oc-mirror プラグイン v2 のバージョンを表示します。
--workspace	リソースと内部アーティファクトが生成される文字列 oc-mirror プラグイン v2 ワークスペースを決定します。

- oc-mirror が生成したリソースを使用するためのクラスター設定

第5章 ASSISTED INSTALLER を使用した ALIBABA CLOUD へのインストール

5.1. ALIBABA CLOUD でのクラスターのアンインストール

Alibaba Cloud にデプロイしたクラスターを削除できます。

5.1.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第6章 AWS へのインストール

6.1. インストール方法

installer-provisioned infrastructure または user-provisioned infrastructure を使用して、Amazon Web Services (AWS) に OpenShift Container Platform をインストールできます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。OpenShift Container Platform をシングルノードにインストールすることもできます。これは、エッジコンピューティング環境に最適な特殊なインストール方法です。

6.1.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる AWS インフラストラクチャーに、クラスターをインストールできます。

- **クラスターの AWS へのクイックインストール**: OpenShift Container Platform インストールプログラムでプロビジョニングされる AWS インフラストラクチャーに OpenShift Container Platform をインストールできます。デフォルトの設定オプションを使用して、クラスターを迅速にインストールできます。

- **カスタマイズされたクラスターの AWS へのインストール**: インストールプログラムがプロビジョニングする AWS インフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の多くのカスタマイズオプションは、インストール後 に利用できます。

- **ネットワークのカスタマイズを使用したクラスターの AWS へのインストール**: インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。

- **ネットワークが制限された環境での AWS へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、インストーラーでプロビジョニングされる AWS インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。

- **クラスターの既存の Virtual Private Cloud へのインストール**: OpenShift Container Platform を既存の AWS Virtual Private Cloud (VPC) にインストールできます。このインストール方法は、新規アカウントまたはインフラストラクチャーを作成する際の制限など、会社のガイドラインによる制約がある場合に使用できます。

- **プライベートクラスターの既存の VPC へのインストール**: プライベートクラスターを既存の AWS VPC にインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

- **クラスターの AWS の government またはシークレットリージョンへのインストール**: OpenShift Container Platform は、機密ワークロードをクラウドで実行する必要のある連邦、州、地方の米国の各種の政府機関、請負業者、教育機関、およびその他の米国の顧客向けに設計されている AWS リージョンにデプロイできます。

6.1.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、独自にプロビジョニングする AWS インフラストラクチャーにクラスターをインストールできます。

- **クラスターの各自でプロビジョニングする AWS インフラストラクチャーへのインストール**: OpenShift Container Platform を、プロビジョニングする AWS インフラストラクチャーにインストールできます。提供される CloudFormation テンプレートを使用して、OpenShift Container Platform インストールに必要な各コンポーネントを表す AWS リソースのスタックを作成できます。

- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したネットワークが制限された環境での AWS へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、独自に提供する AWS インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。ミラーリングされたコンテンツを使用して OpenShift Container Platform をインストールすることは可能ですが、クラスターが AWS API を使用するにはインターネットへのアクセスが必要です。

6.1.3. 単一ノードへのクラスターのインストール

OpenShift Container Platform を単一ノードにインストールすると、高可用性および大規模なクラスターの一部の要件が軽減されます。ただし、シングルノードにインストールするための要件 と、クラウドプロバイダーにシングルノードの OpenShift をインストールするための追加要件 に対処する必要があります。単一ノードのインストールの要件に対処した後、AWS へのカスタマイズされたクラスターのインストール 手順を使用してクラスターをインストールします。単一ノード OpenShift の手動インストール セクションには、OpenShift Container Platform クラスターを単一ノードにインストールする場合の例示的な **install-config.yaml** ファイルが含まれています。

6.1.4. 関連情報

- インストールプロセス

6.2. AWS アカウントの設定

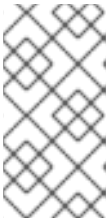
OpenShift Container Platform をインストールする前に、Amazon Web Services (AWS) アカウントを設定する必要があります。

6.2.1. Route 53 の設定

OpenShift Container Platform をインストールするには、使用する Amazon Web Services (AWS) アカウントに、Route 53 サービスの専用のパブリックホストゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。Route 53 サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、AWS または他のソースから新規のものを取得できます。



注記

AWS で新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかります。AWS 経由でドメインを購入する方法についての詳細は、AWS ドキュメントの Registering Domain Names Using Amazon Route 53 を参照してください。

2. 既存のドメインおよびレジストラーを使用している場合、その DNS を AWS に移行します。AWS ドキュメントの Making Amazon Route 53 the DNS Service for an Existing Domain を参照してください。

3. ドメインまたはサブドメインのパブリックホストゾーンを作成します。AWS ドキュメントの Creating a Public Hosted Zone を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。AWS ドキュメントの Getting the Name Servers for a Public Hosted Zone を参照してください。

5. ドメインが使用する AWS Route 53 ネームサーバーのレジストラーレコードを更新します。たとえば、別のアカウントを使用してドメインを Route 53 サービスに登録している場合は、AWS ドキュメントの Adding or Changing Name Servers or Glue Records のトピックを参照してください。

6. サブドメインを使用している場合は、その委任レコードを親ドメインに追加します。これにより、サブドメインの Amazon Route 53 の責任が付与されます。親ドメインの DNS プロバイダーによって要約された委任手順に従います。ハイレベルの手順の例については、AWS ドキュメントの Creating a subdomain that uses Amazon Route 53 as the DNS service without migrating the parent domain を参照してください。

6.2.1.1. AWS Route 53 の Ingress Operator エンドポイント設定

Amazon Web Services (AWS) GovCloud (US) US-West または US-East リージョンのいずれかにインストールする場合、Ingress Operator は Route53 およびタグ付けする API クライアントに **us-gov-west-1** リージョンを使用します。

Ingress Operator は、タグ付けするエンドポイントが 文字列 'us-gov-east-1' を含むように設定される場合、タグ付けする API エンドポイントとして **https://tagging.us-gov-west-1.amazonaws.com** を使用します。

AWS GovCloud (US) エンドポイントについての詳細は、GovCloud (US) についての AWS ドキュメントの Service Endpoints を参照してください。

重要

us-gov-east-1 リージョンにインストールする場合、プライベート、非接続インストールは AWS GovCloud ではサポートされません。

Route 53 設定の例

```
platform:
  aws:
    region: us-gov-west-1
    serviceEndpoints:
    - name: ec2
```

```
    url: https://ec2.us-gov-west-1.amazonaws.com
  - name: elasticloadbalancing
    url: https://elasticloadbalancing.us-gov-west-1.amazonaws.com
  - name: route53
    url: https://route53.us-gov.amazonaws.com 1
  - name: tagging
    url: https://tagging.us-gov-west-1.amazonaws.com 2
```

1 Route 53 は、AWS GovCloud (US) リージョンの両方で **https://route53.us-gov.amazonaws.com** にデフォルト設定されます。

2 US-West リージョンのみにタグ付けするためのエンドポイントがあります。クラスターが別のリージョンにある場合は、このパラメーターを省略します。

6.2.2. AWS アカウントの制限

OpenShift Container Platform クラスターは数多くの Amazon Web Services (AWS) コンポーネントを使用し、デフォルトの サービス制限 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。特定のクラスター設定を使用し、クラスターを特定の AWS リージョンにデプロイするか、アカウントを使用して複数のクラスターを実行する場合、AWS アカウントの追加リソースを要求することが必要になる場合があります。

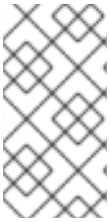
以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある AWS コンポーネントの制限を要約しています。

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明
インスタンスの制限	変動あり。	変動あり。	デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンノード ● 3 つのワーカーノード これらのインスタンスタイプの数は、新規アカウントのデフォルト制限内の値です。追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、異なるインスタンスタイプを使用するには、アカウントの制限を見直し、クラスターが必要なマシンをデプロイできることを確認します。 ほとんどのリージョンでは、ワーカーマシンは **m6i.large** インスタンスを使用し、ブートストラップおよびコントロールプレーンマシンは **m6i.xlarge** インスタンスを使用します。これらのインスタンスタイプをサポートしないすべてのリージョンを含む一部のリージョンでは、**m5.large** および **m5.xlarge** インスタンスが代わりに使用されます。
Elastic IP (EIP)	0 - 1	アカウントごとに 5 つの EIP	クラスターを高可用性設定でプロビジョニングするために、インストールプログラムはそれぞれの リージョン内のアベイラビリティーゾーン にパブリックおよびプライベートのサブネットを作成します。各プライベートサブネットには NAT ゲートウェイ が必要であり、各 NAT ゲートウェイには別個の Elastic IP が必要です。AWS リージョンマップ を確認して、各リージョンにあるアベイラビリティーゾーンの数を判別します。デフォルトの高可用性を利用するには、少なくとも 3 つのアベイラビリティーゾーンがあるリージョンにクラスターをインストールします。アベイラビリティーゾーンが 6 つ以上あるリージョンにクラスターをインストールするには、EIP 制限を引き上げる必要があります。  **重要** **us-east-1** リージョンを使用するには、アカウントの EIP 制限を引き上げる必要があります。

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明
Virtual Private Cloud (VPC)	5	リージョンごとに 5 つの VPC	各クラスターは独自の VPC を作成します。
Elastic Load Balancing (ELB/NLB)	3	リージョンごとに 20	デフォルトで、各クラスターは、マスター API サーバーの内部および外部のネットワークロードバランサーおよびルーターの単一の Classic Elastic Load Balancer を作成します。追加の Kubernetes **Service** オブジェクトをタイプ **LoadBalancer** を指定してデプロイすると、追加のロードバランサーが作成されます。
NAT ゲートウェイ	5	アベイラビリティゾーンごとに 5 つ	クラスターは各アベイラビリティーゾーンに 1 つの NAT ゲートウェイをデプロイします。
Elastic Network Interface (ENI)	12 以上	リージョンごとに 350	デフォルトのインストールは 21 の ENI を作成し、リージョンの各アベイラビリティーゾーンに 1 つの ENI を作成します。たとえば、**us-east-1** リージョンには 6 つのアベイラビリティーゾーンが含まれるため、そのゾーンにデプロイされるクラスターは 27 の ENI を使用します。AWS リージョンマップ を確認して、各リージョンにあるアベイラビリティーゾーンの数を判別します。 クラスターの使用量やデプロイされたワークロード別に作成された追加のマシンや ELB ロードバランサーに対して、追加の ENI が作成されます。
VPC ゲートウェイ	20	アカウントごとに 20	各クラスターは、S3 アクセス用の単一の VPC ゲートウェイを作成します。
S3 バケット	99	アカウントごとに 100 バケット	インストールプロセスでは 1 つの一時的なバケットを作成し、各クラスターのレジストリーコンポーネントがバケットを作成するため、AWS アカウントごとに 99 の OpenShift Container Platform クラスターのみを作成できます。
セキュリティーグループ	250	アカウントごとに 2,500	各クラスターは、10 の個別のセキュリティーグループを作成します。

6.2.3. IAM ユーザーに必要な AWS パーミッション



注記

ベースクラスターリソースを削除するには、IAM ユーザーが領域 **us-east-1** にアクセス許可 **tag:GetResources** を持っている必要があります。AWS API 要件の一部として、OpenShift Container Platform インストールプログラムはこのリージョンでさまざまなアクションを実行します。

AdministratorAccess ポリシーを、Amazon Web Services (AWS) で作成する IAM ユーザーに割り当てる場合、そのユーザーには必要なパーミッションすべてを付与します。OpenShift Container Platform クラスターのすべてのコンポーネントをデプロイするために、IAM ユーザーに以下のパーミッションが必要になります。

例6.1 インストールに必要な EC2 パーミッション

- **ec2:AttachNetworkInterface**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInternetGateways**

- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribePublicIpv4Pools** (**publicIpv4Pool** が **install-config.yaml**で指定されている場合にのみ必要です)
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroupRules**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:DisassociateAddress** (**publicIpv4Pool** が **install-config.yaml**で指定されている場合にのみ必要)
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

例6.2 インストール時のネットワークリソースの作成に必要なパーミッション

- **ec2:AllocateAddress**

- **ec2:AssociateAddress**
- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**

注記

既存の Virtual Private Cloud (VPC) を使用する場合、アカウントではネットワークリソースの作成にこれらのパーミッションを必要としません。

例6.3 インストールに必要な Elastic Load Balancing (ELB) のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**

- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**
- **elasticloadbalancing:SetSecurityGroups**

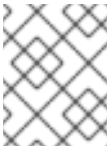
重要

OpenShift Container Platform は、ELB と ELBv2 API サービスの両方を使用してロードバランサーをプロビジョニングします。パーミッションリストには、両方のサービスに必要なパーミッションが表示されます。AWS Web コンソールには、両方のサービスが同じ **elasticloadbalancing** アクション接頭辞を使用しているにもかかわらず、同じアクションを認識しないという既知の問題が存在します。サービスが特定の **elasticloadbalancing** アクションを認識しないという警告は無視できます。

例6.4 インストールに必要な IAM パーミッション

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**

- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagInstanceProfile**
- **iam:TagRole**



注記

AWS アカウントにロードバランサーを作成していない場合、IAM ユーザーには **iam:CreateServiceLinkedRole** パーミッションも必要です。

例6.5 インストールに必要な Route 53 パーミッション

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

例6.6 インストールに必要な Amazon Simple Storage Service (S3) パーミッション

- **s3:CreateBucket**

- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketPolicy**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

例6.7 クラスター Operator が必要とする S3 パーミッション

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

例6.8 ベースクラスターリソースの削除に必要なパーミッション

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeletePlacementGroup**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

例6.9 ネットワークリソースの削除に必要なパーミッション

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**

- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**

注記

既存の VPC を使用する場合、アカウントではネットワークリソースの削除にこれらのパーミッションを必要としません。代わりに、アカウントではネットワークリソースの削除に **tag:UntagResources** パーミッションのみが必要になります。

例6.10 カスタムキー管理サービス (KMS) キーを使用してクラスターをインストールするためのオプションの権限

- **kms:CreateGrant**
- **kms:Decrypt**
- **kms:DescribeKey**
- **kms:Encrypt**
- **kms:GenerateDataKey**
- **kms:GenerateDataKeyWithoutPlainText**
- **kms:ListGrants**
- **kms:RevokeGrant**

例6.11 共有インスタンスロールが割り当てられたクラスターを削除するために必要なパーミッション

- **iam:UntagRole**

例6.12 マニフェストの作成に必要な追加の IAM および S3 パーミッション

- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **s3:AbortMultipartUpload**
- **s3:GetBucketPublicAccessBlock**
- **s3:ListBucket**
- **s3:ListBucketMultipartUploads**

- **s3:PutBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**

注記

クラウドプロバイダーのクレデンシャルをミントモードで管理している場合に、IAM ユーザーには **iam:CreateAccessKey** と **iam:CreateUser** 権限も必要です。

例6.13 インスタンスのオプションのパーミッションおよびインストールのクォータチェック

- **ec2:DescribeInstanceTypeOfferings**
- **servicequotas:ListAWSDefaultServiceQuotas**

例6.14 共有 VPC にクラスターをインストールする場合のクラスター所有者アカウントのオプションの権限

- **sts:AssumeRole**

例6.15 インストールに独自のパブリック IPv4 アドレス(BYOIP)機能を有効にするために必要な権限

- **ec2:DescribePublicIpv4Pools**
- **ec2:DisassociateAddress**

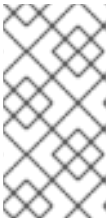
6.2.4. IAM ユーザーの作成

各 Amazon Web Services (AWS) アカウントには、アカウントの作成に使用するメールアドレスに基づく root ユーザーアカウントが含まれます。これは高度な権限が付与されたアカウントであり、初期アカウントにのみ使用し、請求設定また初期のユーザーセットの作成およびアカウントのセキュリティー保護のために使用することが推奨されています。

OpenShift Container Platform をインストールする前に、セカンダリー IAM 管理ユーザーを作成します。AWS ドキュメントの Creating an IAM User in Your AWS Account 手順を実行する際に、以下のオプションを設定します。

手順

1. IAM ユーザー名を指定し、**Programmatic access** を選択します。
2. **AdministratorAccess** ポリシーを割り当て、アカウントにクラスターを作成するために十分なパーミッションがあることを確認します。このポリシーはクラスターに対し、各 OpenShift Container Platform コンポーネントに認証情報を付与する機能を提供します。クラスターはコンポーネントに対し、それらが必要とする認証情報のみを付与します。



注記

必要なすべての AWS パーミッションを付与し、これをユーザーに割り当てるポリシーを作成することは可能ですが、これは優先されるオプションではありません。クラスターには追加の認証情報を個別コンポーネントに付与する機能がないため、同じ認証情報がすべてのコンポーネントによって使用されます。

3. オプション: タグを割り当て、メタデータをユーザーに追加します。
4. 指定したユーザー名に **AdministratorAccess** ポリシーが付与されていることを確認します。
5. アクセスキー ID およびシークレットアクセスキーの値を記録します。ローカルマシンをインストールプログラムを実行するように設定する際にこれらの値を使用する必要があります。

重要

クラスターのデプロイ時に、マルチファクター認証デバイスの使用中に生成した一時的なセッショントークンを使用して AWS に対する認証を行うことはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。

6.2.5. IAM ポリシーと AWS 認証

デフォルトでは、インストールプログラムは、ブートストラップ、コントロールプレーン、およびコンピュートインスタンスのインスタンスプロファイルを作成し、クラスターの動作に必要な権限を付与します。

ただし、独自の IAM ロールを作成して、インストールプロセスの一部として指定できます。クラスターをデプロイするため、またはインストール後にクラスターを管理するために、独自のロールを指定する必要がある場合があります。以下に例を示します。

- 組織のセキュリティーポリシーでは、より制限的なアクセス許可セットを使用してクラスターをインストールする必要があります。
- インストール後、クラスターは、追加サービスへのアクセスを必要とする Operator で設定されます。

独自の IAM ロールを指定する場合は、次の手順を実行できます。

- デフォルトのポリシーから始めて、必要に応じて調整します。詳細については、「IAM インスタンスプロファイルのデフォルトのアクセス許可」を参照してください。
- AWS IAM Access Analyzer (Identity and Access Management Access Analyzer) を使用して、クラスターのアクティビティーに基づくポリシーテンプレートを作成します。詳細は、「AWS IAM Analyzer を使用してポリシーテンプレートの作成」を参照してください。

6.2.5.1. IAM インスタンスプロファイルのデフォルトのアクセス許可

デフォルトでは、インストールプログラムは、ブートストラップ、コントロールプレーン、およびワーカーインスタンスの IAM インスタンスプロファイルを作成し、クラスターの動作に必要な権限を付与します。

次のリストでは、コントロールプレーンとコンピュートマシンのデフォルトのアクセス許可を指定します。

-

例6.16 コントロールプレーンインスタンスのプロファイル向けデフォルト IAM ロールのパーミッション

- ec2:AttachVolume
- ec2:AuthorizeSecurityGroupIngress
- ec2:CreateSecurityGroup
- ec2:CreateTags
- ec2:CreateVolume
- ec2:DeleteSecurityGroup
- ec2:DeleteVolume
- ec2:Describe*
- ec2:DetachVolume
- ec2:ModifyInstanceAttribute
- ec2:ModifyVolume
- ec2:RevokeSecurityGroupIngress
- elasticloadbalancing:AddTags
- elasticloadbalancing:AttachLoadBalancerToSubnets
- elasticloadbalancing:ApplySecurityGroupsToLoadBalancer
- elasticloadbalancing:CreateListener
- elasticloadbalancing:CreateLoadBalancer
- elasticloadbalancing:CreateLoadBalancerPolicy
- elasticloadbalancing:CreateLoadBalancerListeners
- elasticloadbalancing:CreateTargetGroup
- elasticloadbalancing:ConfigureHealthCheck
- elasticloadbalancing:DeleteListener
- elasticloadbalancing:DeleteLoadBalancer
- elasticloadbalancing:DeleteLoadBalancerListeners
- elasticloadbalancing:DeleteTargetGroup
- elasticloadbalancing:DeregisterInstancesFromLoadBalancer
- elasticloadbalancing:DeregisterTargets
- elasticloadbalancing:Describe*

- **elasticloadbalancing:DetachLoadBalancerFromSubnets**
- **elasticloadbalancing:ModifyListener**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**
- **kms:DescribeKey**

例6.17 コンピュートインスタンスプロファイル向けデフォルト IAM ロールのパーミッション

- **ec2:DescribeInstances**
- **ec2:DescribeRegions**

6.2.5.2. 既存の IAM ロールの指定

インストールプログラムがデフォルトのアクセス許可で IAM インスタンスプロファイルを作成できるようにする代わりに、**install-config.yaml** ファイルを使用して、コントロールプレーンとコンピュートインスタンスの既存の IAM ロールを指定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

1. コントロールプレーンマシンの既存のロールで **compute.platform.aws.iamRole** を更新します。

   コンピュートインスタンスの IAM ロールを含む install-config.yaml ファイルのサンプル

   ```
   compute:
   - hyperthreading: Enabled
     name: worker
     platform:
       aws:
         iamRole: ExampleRole
   ```

2. コンピュートマシンの既存のロールで **controlPlane.platform.aws.iamRole** を更新します。

コントロールプレーンインスタンスの IAM ロールを含む install-config.yaml ファイルのサンプル

```
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      iamRole: ExampleRole
```

3. ファイルを保存し、OpenShift Container Platform クラスターのインストール時に参照します。

関連情報

- クラスターのデプロイ

6.2.5.3. AWS IAM Analyzer を使用してポリシーテンプレートの作成

コントロールプレーンとコンピュートインスタンスプロファイルに必要な最小限のアクセス許可セットは、クラスターが日常の運用のためにどのように設定されているかによって異なります。

クラスターインスタンスに必要なアクセス許可を決定する 1 つの方法は、IAM Access Analyzer (AWS Identity and Access Management Access Analyzer) を使用してポリシーテンプレートを作成することです。

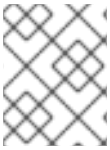
- ポリシーテンプレートには、クラスターが指定された期間に使用したアクセス許可が含まれています。
- その後、テンプレートを使用して、きめ細かい権限を持つポリシーを作成できます。

手順

全体的なプロセスは次のようになります。

1. CloudTrail が有効になっていることを確認します。CloudTrail は、ポリシーテンプレートの作成に必要な API 呼び出しを含め、AWS アカウントのすべてのアクションとイベントを記録します。詳細は、CloudTrail の操作 に関する AWS ドキュメントを参照してください。
2. コントロールプレーンインスタンスのインスタンスプロファイルとコンピューティングインスタンスのインスタンスプロファイルを作成します。PowerUserAccess などの寛容なポリシーを各ロールに割り当ててください。詳細は、インスタンスプロファイルロールの作成 に関する AWS ドキュメントを参照してください。
3. クラスターを開発環境にインストールし、必要に応じて設定します。クラスターが本番環境でホストするすべてのアプリケーションを必ずデプロイしてください。
4. クラスターを徹底的にテストします。クラスターをテストすると、必要なすべての API 呼び出しがログに記録されることが保証されます。
5. IAM Access Analyzer を使用して、各インスタンスプロファイルのポリシーテンプレートを作成します。詳細は、CloudTrail ログに基づいてポリシーを生成する ための AWS ドキュメントを参照してください。
6. きめ細かいポリシーを作成し、各インスタンスプロファイルに追加します。

7. 各インスタンスプロファイルから許容ポリシーを削除します。
8. 新しいポリシーで既存のインスタンスプロファイルを使用して実稼働クラスターをデプロイします。



注記

ポリシーに IAM 条件 を追加して、ポリシーをより制限し、組織のセキュリティー要件に準拠させることができます。

6.2.6. サポートされている AWS Marketplace リージョン

北米でオファーを購入したお客様は、AWS Marketplace イメージを使用して、OpenShift Container Platform クラスターをインストールすることができます。

このオファーは北米で購入する必要がありますが、以下のサポートされているパーティションのいずれかにクラスターをデプロイできます。

- 公開
- GovCloud

注記

AWS Marketplace イメージを使用した OpenShift Container Platform クラスターのデプロイは、AWS シークレットリージョンまたは中国リージョンではサポートされていません。

6.2.7. サポートされている AWS リージョン

OpenShift Container Platform クラスターを以下のリージョンにデプロイできます。

注記

ベースクラスターリソースを削除するには、IAM ユーザーが領域 **us-east-1** にアクセス許可 **tag:GetResources** を持っている必要があります。AWS API 要件の一部として、OpenShift Container Platform インストールプログラムはこのリージョンでさまざまなアクションを実行します。

6.2.7.1. AWS パブリックリージョン

以下の AWS パブリックリージョンがサポートされます。

- **af-south-1** (Cape Town)
- **ap-east-1** (Hong Kong)
- **ap-northeast-1** (Tokyo)
- **ap-northeast-2** (Seoul)
- **ap-northeast-3** (Osaka)
- **ap-south-1** (Mumbai)

- **ap-south-2** (Hyderabad)
- **ap-southeast-1** (Singapore)
- **ap-southeast-2** (Sydney)
- **ap-southeast-3** (Jakarta)
- **ap-southeast-4** (Melbourne)
- **ca-central-1** (Central)
- **ca-west-1** (Calgary)
- **eu-central-1** (Frankfurt)
- **eu-central-2** (Zurich)
- **eu-north-1** (Stockholm)
- **eu-south-1** (Milan)
- **eu-south-2** (Spain)
- **eu-west-1** (Ireland)
- **eu-west-2** (London)
- **eu-west-3** (Paris)
- **il-central-1** (Tel Aviv)
- **me-central-1** (UAE)
- **me-south-1** (Bahrain)
- **sa-east-1** (São Paulo)
- **us-east-1** (N. Virginia)
- **us-east-2** (Ohio)
- **us-west-1** (N. California)
- **us-west-2** (Oregon)

6.2.7.2. AWS GovCloud リージョン

以下の AWS GovCloud リージョンがサポートされます。

- **us-gov-west-1**
- **us-gov-east-1**

6.2.7.3. AWS SC2S および C2S シークレットリージョン

以下の AWS シークレットリージョンがサポートされています。

- **us-isob-east-1** Secret Commercial Cloud Services (SC2S)
- **us-iso-east-1** Commercial Cloud Services (C2S)

6.2.7.4. AWS China リージョン

以下の AWS China リージョンがサポートされます。

- **cn-north-1** (Beijing)
- **cn-northwest-1** (Ningxia)

6.2.8. 次のステップ

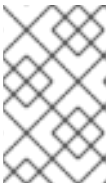
- OpenShift Container Platform クラスターをインストールします。
  - インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトオプションを使用した クラスターのクイックインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのクラウドのカスタマイズを使用したクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのネットワークのカスタマイズを使用したクラスターのインストール
  - CloudFormation テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール

6.3. INSTALLER-PROVISIONED INFRASTRUCTURE

6.3.1. AWS にクラスターをインストールする準備

以下の手順を実行して、AWS に OpenShift Container Platform クラスターをインストールする準備をします。

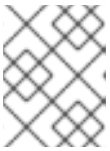
- クラスターのインターネット接続を検証します。
- AWS アカウントを設定します。
- インストールプログラムをダウンロードします。



注記

非接続環境にインストールする場合は、ミラーリングしたコンテンツからインストールプログラムを抽出します。詳細は、非接続インストール用のイメージのミラーリング を参照してください。

- OpenShift CLI (**oc**) をインストールします。



注記

非接続環境にインストールする場合は、ミラーホストに **oc** をインストールします。

- SSH キーペアを生成します。OpenShift Container Platform クラスターのデプロイ後にこのキーペアを使用して、クラスターのノードに対する認証を行うことができます。
- クラウドアイデンティティーおよびアクセス管理 (IAM) API が環境内でアクセスできない場合、または管理者レベルの認証情報シークレットを **kube-system** namespace に保存しない場合は、AWS の長期認証情報を手動で作成 するか、Amazon Web Services Security Token Service (AWS STS) で 短期認証情報を使用するように AWS クラスターを設定 します。

6.3.1.1. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

6.3.1.2. インストールプログラムの取得

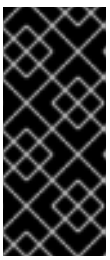
OpenShift Container Platform をインストールする前に、インストールファイルをミラーホストにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

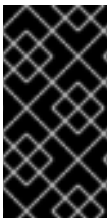
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

6.3.1.3. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。非接続環境でのクラスターを更新する場合は、更新する予定の **oc** バージョンをインストールします。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

6.3.1.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの ~/**.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

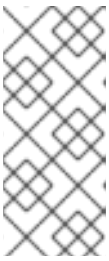
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.3.1.5. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

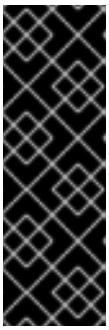
- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

6.3.2. AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、デフォルトの設定オプションを使用するクラスターを Amazon Web Services (AWS) にインストールできます。

6.3.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

6.3.2.2. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

ディレクトリーを指定する場合:

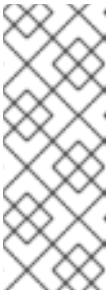
- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. プロンプト時に値を指定します。
   a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

   b. ターゲットに設定するプラットフォームとして **aws** を選択します。
   c. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。



注記

AWS アクセスキー ID およびシークレットアクセスキーは、インストールホストの現行ユーザーのホームディレクトリーの ~/**.aws/credentials** に保存されます。エクスポートされたプロファイルの認証情報がファイルにない場合は、インストールプログラムにより認証情報の入力が求めるプロンプトが出されます。インストールプログラムに指定する認証情報は、ファイルに保存されます。

d. クラスターのデプロイ先とする AWS リージョンを選択します。

e. クラスターに設定した Route 53 サービスのベースドメインを選択します。

f. クラスターの記述名を入力します。

g. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

3. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。

注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

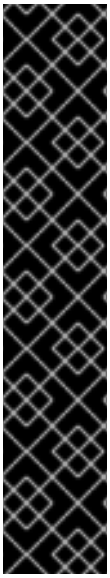
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

関連情報

- AWS プロファイルおよび認証情報の設定についての詳細は、AWS ドキュメントの Configuration and credential file settings を参照してください。

6.3.2.3. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.2.4. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

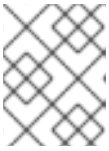
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

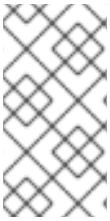
6.3.2.5. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

6.3.3. カスタマイズによる AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが Amazon Web Services (AWS) 上にプロビジョニングするインフラストラクチャーにカスタマイズしたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

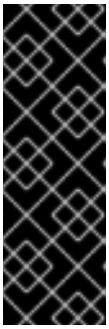


注記

OpenShift Container Platform インストール設定のスコープは意図的に狭められています。単純さを確保し、確実にインストールを実行できるように設計されているためです。インストールが完了した後にさらに多くの OpenShift Container Platform 設定タスクを実行することができます。

6.3.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

6.3.3.2. AWS Marketplace イメージの取得

AWS Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に AWS を通じてサブスクライブする必要があります。オファーにサブスクライブすると、インストールプログラムがコンピュートノードのデプロイに使用する AMI ID が提供されます。

前提条件

- オファーを購入するための AWS アカウントを持っている。このアカウントは、クラスターのインストールに使用されるアカウントと同じである必要はありません。

手順

1. AWS Marketplace で OpenShift Container Platform サブスクリプションを完了します。

2. ご使用の AWS リージョンの AMI ID を記録します。インストールプロセスの一環として、クラスターをデプロイする前に、この値で **install-config.yaml** ファイルを更新する必要があります。

AWS Marketplace コンピュートノードを含む install-config.yaml ファイルのサンプル

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) AWS Marketplace サブスクリプションの AMI ID。

(2) AMI ID は特定の AWS リージョンに関連付けられています。インストール設定ファイルを作成するときは、サブスクリプションの設定時に指定したものと同じ AWS リージョンを選択してください。

6.3.3.3. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> (1)
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

注記

3 ノードクラスターをインストールする場合は、必ず **compute.replicas** パラメーターを **0** に設定してください。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細については、「AWS に 3 ノードクラスターをインストールする」を参照してください。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. クラスターをインストールする VPC のサブネットを定義します。

```
subnets:
- subnet-1
- subnet-2
- subnet-3
```

d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

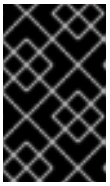
e. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- AWS のインストール設定パラメーター

6.3.3.3.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

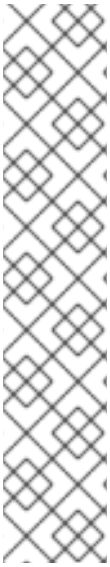
表6.1 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

6.3.3.3.2. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services(AWS)インスタンスタイプは OpenShift Container Platform でテストされています。

注記

AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

例6.18 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***

- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

6.3.3.3.3. 64 ビット ARM インフラストラクチャー上の AWS のテスト済みインスタンスタイプ

次の Amazon Web Services (AWS) 64 ビット ARM インスタンスタイプは、OpenShift Container Platform でテストされています。

注記

AWS ARM インスタンスには、次のチャートに含まれるマシンタイプを使用してください。チャートに記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、クラスターインストールの最小リソース要件に記載されている最小リソース要件と一致していることを確認してください。

例6.19 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

6.3.3.3.4. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

インストール設定ファイル **install-config.yaml** をカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
```

```
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 6
      metadataService:
        authentication: Optional 7
      type: m6i.xlarge
  replicas: 3
compute: 8
- hyperthreading: Enabled 9
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 10
      metadataService:
        authentication: Optional 11
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster 12
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 13
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 14
    propagateUserTags: true 15
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 16
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-0c5d3e03c0ab9b19a 17
    serviceEndpoints: 18
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 19
fips: false 20
sshKey: ssh-ed25519 AAAA... 21
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 22
```

```
additionalTrustBundle: | 23
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
imageContentSources: 24
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 12 14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 8 15 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 9 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 10 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

7 11 Amazon EC2 Instance Metadata Service v2 (IMDSv2) を要求するかどうか。IMDSv2 を要求するには、パラメーター値を **Required** に設定します。IMDSv1 と IMDSv2 の両方の使用を許可するには、パラメーター値を **Optional** に設定します。値が指定されていない場合、IMDSv1 と IMDSv2 の両方が許可されます。

注記

クラスターのインストール中に設定されるコントロールプレーンマシンの IMDS 設定は、AWS CLI を使用してのみ変更できます。コンピュートマシンの IMDS 設定は、コンピュートマシンセットを使用して変更できます。

13 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

16 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

17 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

18 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

19 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

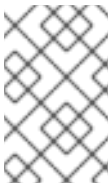
20 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

21 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

22 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

23 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

24 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

6.3.3.3.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキ

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。
4.

指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

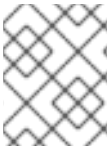
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.3.3.4. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように AWS クラスターを設定する の手順に従ってください。

6.3.3.4.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

   サンプル CredentialsRequest オブジェクト

   ```
   apiVersion: cloudcredential.openshift.io/v1
   kind: CredentialsRequest
   metadata:
     name: <component_credentials_request>
     namespace: openshift-cloud-credential-operator
     ...
   spec:
   ```

```
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

6.3.3.4.2. 短期認証情報を使用するように AWS クラスターを設定

AWS Security Token Service (STS) を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な AWS リソースを作成する必要があります。

6.3.3.4.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

> **例6.20 必要な AWS パーミッション**
>
> **必要な iam 権限**
>
> - **iam:CreateOpenIDConnectProvider**
> - **iam:CreateRole**
> - **iam:DeleteOpenIDConnectProvider**
> - **iam:DeleteRole**
> - **iam:DeleteRolePolicy**
> - **iam:GetOpenIDConnectProvider**
> - **iam:GetRole**
> - **iam:GetUser**
> - **iam:ListOpenIDConnectProviders**
> - **iam:ListRolePolicies**
> - **iam:ListRoles**
> - **iam:PutRolePolicy**

- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

必要な s3 権限

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

必要な cloudfront 権限

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

例6.21 CloudFront を使用したプライベート S3 バケットに対する追加の権限

- **cloudfront:CreateCloudFrontOriginAccessIdentity**
- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**

- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

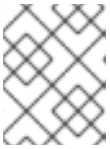
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

6.3.3.4.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

AWS リソースを作成するときは、次のオプションがあります。

- **ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。これはリソースを作成する最も簡単な方法です。単一コマンドでの AWS リソースの作成 を参照してください。
- AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。AWS リソースの個別の作成 を参照してください。

6.3.3.4.2.2.1. 単一コマンドでの AWS リソースの作成

ccoctl ツールが AWS リソースの作成に使用するプロセスが組織の要件を自動的に満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。詳細は、「AWS リソースの個別の作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

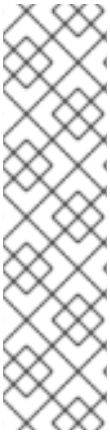
```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.3.4.2.2.2. AWS リソースの個別の作成

ccoctl ツールを使用して、AWS リソースを個別に作成できます。このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。詳細は、「単一コマンドによる AWS リソースの作成」を参照してください。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. 次のコマンドを実行して、クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. 次のコマンドを実行して、AWS 上に OpenID Connect ID プロバイダーと S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1. **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
3. **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

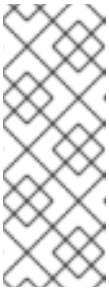
   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   c. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

   ```
   $ ccoctl aws create-iam-roles \
     --name=<name> \
     --region=<aws_region> \
   ```

```
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```



注記

GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.3.4.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

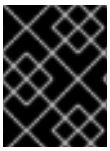
   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

6.3.3.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

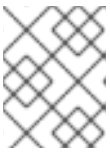
1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。



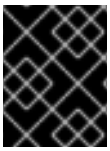
注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

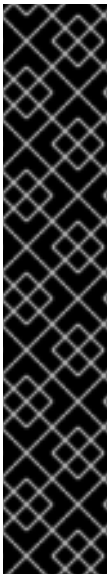


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.3.6. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.3.7. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

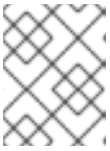
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

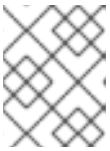


注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

6.3.3.8. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

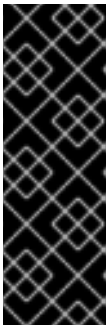
6.3.4. ネットワークのカスタマイズによる AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、カスタマイズしたネットワーク設定オプションを使用して、Amazon Web Services (AWS) にクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

6.3.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

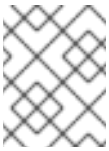
6.3.4.2. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

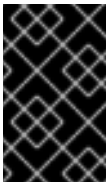
マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。



重要

CIDR 範囲 **172.17.0.0/16** は libVirt によって予約されています。この範囲、またはこの範囲と重複する範囲をクラスター内のネットワークに使用することはできません。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 でネットワークプラグインをさらにカスタマイズできます。

6.3.4.3. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

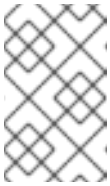
1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. クラスターをインストールする VPC のサブネットを定義します。

```
subnets:
- subnet-1
- subnet-2
- subnet-3
```

d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

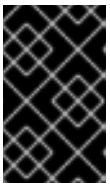
e. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- AWS のインストール設定パラメーター

6.3.4.3.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

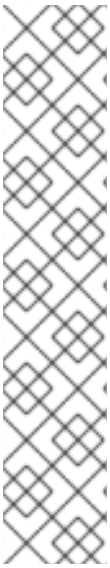
表6.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

6.3.4.3.2. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services(AWS)インスタンスタイプは OpenShift Container Platform でテストされています。

注記

AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

> **例6.22 64 ビット x86 アーキテクチャーに基づくマシンタイプ**

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

6.3.4.3.3. 64 ビット ARM インフラストラクチャー上の AWS のテスト済みインスタンスタイプ

次の Amazon Web Services (AWS) 64 ビット ARM インスタンスタイプは、OpenShift Container Platform でテストされています。

注記

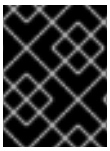
AWS ARM インスタンスには、次のチャートに含まれるマシンタイプを使用してください。チャートに記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、クラスターインストールの最小リソース要件に記載されている最小リソース要件と一致していることを確認してください。

例6.23 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

6.3.4.3.4. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

インストール設定ファイル **install-config.yaml** をカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      metadataService:
        authentication: Optional (7)
      type: m6i.xlarge
  replicas: 3
compute: (8)
- hyperthreading: Enabled (9)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (10)
      metadataService:
        authentication: Optional (11)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (12)
networking: (13)
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (14)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (15)
```

```
    propagateUserTags: true 16
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 17
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-0c5d3e03c0ab9b19a 18
    serviceEndpoints: 19
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 20
fips: false 21
sshKey: ssh-ed25519 AAAA... 22
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 23
additionalTrustBundle: | 24
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: 25
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 12 15 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 8 13 16 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

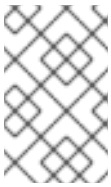
5 9 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 10 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

7 11 Amazon EC2 Instance Metadata Service v2 (IMDSv2) を要求するかどうか。IMDSv2 を要求するには、パラメーター値を **Required** に設定します。IMDSv1 と IMDSv2 の両方の使用を許可するには、パラメーター値を **Optional** に設定します。値が指定されていない場合、IMDSv1 と IMDSv2 の両方が許可されます。



注記

クラスターのインストール中に設定されるコントロールプレーンマシンの IMDS 設定は、AWS CLI を使用してのみ変更できます。コンピュートマシンの IMDS 設定は、コンピュートマシンセットを使用して変更できます。

14 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

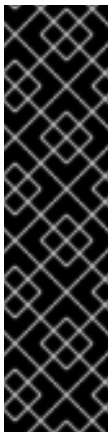
17 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

18 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

19 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

20 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

21 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。



重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

22 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

23 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

24 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

25 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

6.3.4.3.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
```

```
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

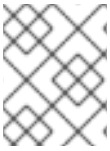
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.3.4.4. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように AWS クラスターを設定する の手順に従ってください。

6.3.4.4.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
   ```

```
--included \ 1
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
--to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

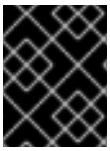
シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
```

```
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```



重要

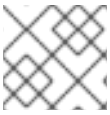
手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

6.3.4.4.2. 短期認証情報を使用するように AWS クラスターを設定

AWS Security Token Service (STS) を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な AWS リソースを作成する必要があります。

6.3.4.4.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

  例6.24 必要な AWS パーミッション

  必要な iam 権限

- iam:CreateOpenIDConnectProvider
- iam:CreateRole
- iam:DeleteOpenIDConnectProvider
- iam:DeleteRole
- iam:DeleteRolePolicy
- iam:GetOpenIDConnectProvider
- iam:GetRole
- iam:GetUser
- iam:ListOpenIDConnectProviders
- iam:ListRolePolicies
- iam:ListRoles
- iam:PutRolePolicy
- iam:TagOpenIDConnectProvider
- iam:TagRole

必要な s3 権限

- s3:CreateBucket
- s3:DeleteBucket
- s3:DeleteObject
- s3:GetBucketAcl
- s3:GetBucketTagging
- s3:GetObject
- s3:GetObjectAcl
- s3:GetObjectTagging
- s3:ListBucket
- s3:PutBucketAcl
- s3:PutBucketPolicy
- s3:PutBucketPublicAccessBlock
- s3:PutBucketTagging
- s3:PutObject

- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

必要な cloudfront 権限

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

例6.25 CloudFront を使用したプライベート S3 バケットに対する追加の権限

- **cloudfront:CreateCloudFrontOriginAccessIdentity**
- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**
- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

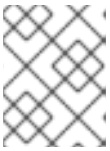
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
```

```
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

6.3.4.4.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

AWS リソースを作成するときは、次のオプションがあります。

- **ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。これはリソースを作成する最も簡単な方法です。単一コマンドでの AWS リソースの作成 を参照してください。
- AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。AWS リソースの個別の作成 を参照してください。

6.3.4.4.2.2.1. 単一コマンドでの AWS リソースの作成

ccoctl ツールが AWS リソースの作成に使用するプロセスが組織の要件を自動的に満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。詳細は、「AWS リソースの個別の作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-all \
  --name=<name> \1
  --region=<aws_region> \2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \3
  --output-dir=<path_to_ccoctl_output_dir> \4
  --create-private-s3-bucket 5
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.4.4.2.2.2. AWS リソースの個別の作成

ccoctl ツールを使用して、AWS リソースを個別に作成できます。このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。詳細は、「単一コマンドによる AWS リソースの作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. 次のコマンドを実行して、クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. 次のコマンドを実行して、AWS 上に OpenID Connect ID プロバイダーと S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1. **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
3. **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

   b. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
```

```
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

2 **install-config.yaml** ファイルの場所を指定します。

3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

c. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

注記

GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
```

```
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.4.4.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

6.3.4.5. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

6.3.4.5.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表6.3 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	型	説明
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表6.4 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表6.5 ovnKubernetesConfig オブジェクト

フィールド	型	説明

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表6.6 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表6.7 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表6.8 **policyAuditConfig** オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表6.9 **gatewayConfig** オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

フィールド	型	説明
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表6.10 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表6.11 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表6.12 **ipsecConfig** オブジェクト

フィールド	型	説明

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

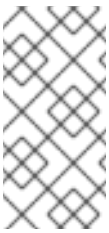
```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

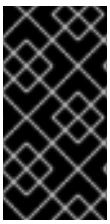
表6.13 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	型	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

6.3.4.6. 高度なネットワーク設定の指定

ネットワークプラグインに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。



重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 次の例のように、**cluster-network-03-config.yml** ファイルでクラスターの高度なネットワーク設定を指定します。

   OVN-Kubernetes ネットワークプロバイダーを有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

注記

AWS で Network Load Balancer (NLB) を使用する方法の詳細は、Network Load Balancer を使用した AWS での Ingress クラスタートラフィックの設定 を参照してください。

6.3.4.7. 新規 AWS クラスターでの Ingress コントローラーネットワークロードバランサーの設定

新規クラスターに AWS Network Load Balancer (NLB) がサポートする Ingress Controller を作成できます。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

新規クラスターの AWS NLB がサポートする Ingress Controller を作成します。

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-ingress-default-ingresscontroller.yaml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

出力例

```
cluster-ingress-default-ingresscontroller.yaml
```

3. エディターで **cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、必要な Operator 設定を記述するカスタムリソース (CR) を入力します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
      providerParameters:
        type: AWS
        aws:
          type: NLB
    type: LoadBalancerService
```

4. **cluster-ingress-default-ingresscontroller.yaml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-ingress-default-ingresscontroller.yaml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

6.3.4.8. OVN-Kubernetes を使用したハイブリッドネットワークの設定

OVN-Kubernetes ネットワークプラグインを使用してハイブリッドネットワークを使用するようにクラスターを設定できます。これにより、異なるノードのネットワーク設定をサポートするハイブリッドクラスターが可能になります。

前提条件

- **install-config.yaml** ファイルで **networking.networkType** パラメーターの **OVNKubernetes** を定義していること。詳細は、選択したクラウドプロバイダーでの OpenShift Container Platform ネットワークのカスタマイズの設定についてのインストールドキュメントを参照してください。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

ここでは、以下のようになります。

<installation_directory>

クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
EOF
```

ここでは、以下のようになります。

<installation_directory>

クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

3. **cluster-network-03-config.yml** ファイルをエディターで開き、以下の例のようにハイブリッドネットワークで OVN-Kubernetes を設定します。

ハイブリッドネットワーク設定の指定

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork: 1
        - cidr: 10.132.0.0/14
          hostPrefix: 23
```

1 追加のオーバーレイネットワーク上のノードに使用される CIDR 設定を指定します。**hybridClusterNetwork** CIDR は **clusterNetwork** CIDR と重複できません。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

6.3.4.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

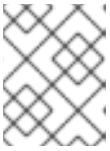
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。



注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
-
```

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.4.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.4.11. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

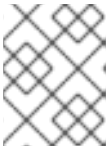
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

6.3.4.12. 次のステップ

- インストールを検証 します。

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

6.3.5. ネットワークが制限された環境での AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、既存の Amazon Virtual Private Cloud (VPC) にインストールリリースコンテンツの内部ミラーを作成することにより、制限付きネットワーク内の Amazon Web Services (AWS) にクラスターをインストールできます。

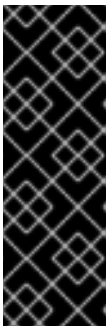
6.3.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- AWS に既存の VPC が必要です。インストーラーでプロビジョニングされるインフラストラクチャーを使用してネットワークが制限された環境にインストールする場合は、インストーラーでプロビジョニングされる VPC を使用することはできません。以下の要件のいずれかを満たすユーザーによってプロビジョニングされる VPC を使用する必要があります。
  - ミラーレジストリーが含まれる。
  - 別の場所でホストされるミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続がある。
- クラスターをホストするために AWS アカウントを設定 している。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

6.3.5.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

6.3.5.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

6.3.5.3. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS) の既存の Amazon Virtual Private Cloud (VPC) における既存サブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

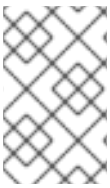
インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

6.3.5.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル

- VPC
- VPC DHCP オプション
- VPC エンドポイント



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- VPC は **kubernetes.io/cluster/.*: owned**、**Name**、**openshift.io/cluster** タグを使用できません。
  インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。**Name** タグは EC2 **Name** フィールドと重複し、その結果インストールが失敗するため、使用できません。

- OpenShift Container Platform クラスターを AWS Outpost に拡張し、既存の Outpost サブネットを使用する場合、既存のサブネットで **kubernetes.io/cluster/unmanaged: true** タグを使用する必要があります。このタグを適用しないと、Cloud Controller Manager が Outpost サブネットにサービスロードバランサーを作成するため、インストールが失敗する可能性があります。これはサポートされていない設定です。

- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドと **platform.aws.hostedZoneRole** フィールドを使用して、ホストゾーンを定義できます。クラスターをインストールするアカウントとプライベートホストゾーンを共有することで、別のアカウントからプライベートホストゾーンを使用できます。別のアカウントからプライベートホストゾーンを使用する場合は、**Passthrough** または **Manual** 認証情報モードを使用する必要があります。

非接続環境で作業している場合、EC2、ELB、および S3 エンドポイントのパブリック IP アドレスに到達することはできません。インストール中にインターネットトラフィックを制限するレベルに応じて、次の設定オプションを使用できます。

オプション 1: VPC エンドポイントを作成する
VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

このオプションを使用すると、VPC および必要な AWS サービスの間でネットワークトラフィックがプライベートのままになります。

オプション 2: VPC エンドポイントなしでプロキシーを作成する
インストールプロセスの一環として、HTTP または HTTPS プロキシーを設定できます。このオプションを使用すると、インターネットトラフィックはプロキシーを経由して、必要な AWS サービスに到達します。

オプション 3: VPC エンドポイントでプロキシーを作成する
インストールプロセスの一環として、VPC エンドポイントを使用して HTTP または HTTPS プロキシーを設定できます。VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

install-config.yaml ファイルでプロキシーを設定するときに、これらのエンドポイントを **noProxy** フィールドに追加します。このオプションを使用すると、プロキシーはクラスターがインターネットに直接アクセスするのを防ぎます。ただし、VPC と必要な AWS サービスの間のネットワークトラフィックはプライベートのままです。

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。

6.3.5.3.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

6.3.5.3.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

6.3.5.3.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

6.3.5.4. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

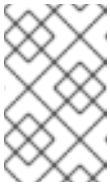
   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. クラスターをインストールする VPC のサブネットを定義します。

```
subnets:
- subnet-1
- subnet-2
- subnet-3
```

d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

e. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- AWS のインストール設定パラメーター

6.3.5.4.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

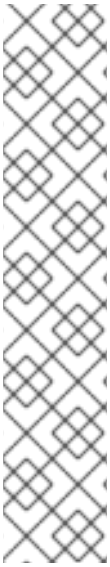
表6.14 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

6.3.5.4.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

インストール設定ファイル **install-config.yaml** をカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
```

```
controlPlane: 3 4
  hyperthreading: Enabled 5
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 6
      metadataService:
        authentication: Optional 7
      type: m6i.xlarge
  replicas: 3
compute: 8
- hyperthreading: Enabled 9
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 10
      metadataService:
        authentication: Optional 11
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster 12
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 13
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 14
    propagateUserTags: true 15
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 16
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-0c5d3e03c0ab9b19a 17
```

```
    serviceEndpoints: 18
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 19
fips: false 20
sshKey: ssh-ed25519 AAAA... 21
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 22
additionalTrustBundle: | 23
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: 24
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 12 14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 8 15 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 9 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 10 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

7 11 Amazon EC2 Instance Metadata Service v2 (IMDSv2) を要求するかどうか。IMDSv2 を要求するには、パラメーター値を **Required** に設定します。IMDSv1 と IMDSv2 の両方の使用を許可するには、パラメーター値を **Optional** に設定します。値が指定されていない場合、IMDSv1 と IMDSv2 の両方が許可されます。

注記

クラスターのインストール中に設定されるコントロールプレーンマシンの IMDS 設定は、AWS CLI を使用してのみ変更できます。コンピュートマシンの IMDS 設定は、コンピュートマシンセットを使用して変更できます。

13 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

16 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

17 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

18 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

19 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

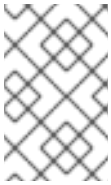
20 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

21 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

22 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

23 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

24 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

6.3.5.4.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に .

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

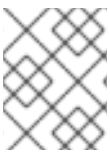
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.3.5.5. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように AWS クラスターを設定する の手順に従ってください。

6.3.5.5.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

   サンプル CredentialsRequest オブジェクト

   -

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

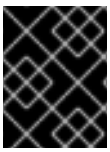
シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
```

```
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

6.3.5.5.2. 短期認証情報を使用するように AWS クラスターを設定

AWS Security Token Service (STS) を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な AWS リソースを作成する必要があります。

6.3.5.5.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

例6.26 必要な AWS パーミッション

必要な iam 権限

- **iam:CreateOpenIDConnectProvider**
- **iam:CreateRole**
- **iam:DeleteOpenIDConnectProvider**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetOpenIDConnectProvider**
- **iam:GetRole**
- **iam:GetUser**
- **iam:ListOpenIDConnectProviders**
- **iam:ListRolePolicies**

- **iam:ListRoles**
- **iam:PutRolePolicy**
- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

必要な s3 権限

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

必要な cloudfront 権限

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

例6.27 CloudFront を使用したプライベート S3 バケットに対する追加の権限

- **cloudfront:CreateCloudFrontOriginAccessIdentity**

- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**
- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。

- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

6.3.5.5.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

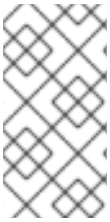
AWS リソースを作成するときは、次のオプションがあります。

- **ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。これはリソースを作成する最も簡単な方法です。単一コマンドでの AWS リソースの作成 を参照してください。
- AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。AWS リソースの個別の作成 を参照してください。

6.3.5.5.2.2.1. 単一コマンドでの AWS リソースの作成

ccoctl ツールが AWS リソースの作成に使用するプロセスが組織の要件を自動的に満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。詳細は、「AWS リソースの個別の作成」を参照してください。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

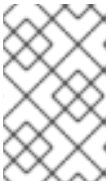
注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.5.5.2.2.2. AWS リソースの個別の作成

ccoctl ツールを使用して、AWS リソースを個別に作成できます。このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。詳細は、「単一コマンドによる AWS リソースの作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. 次のコマンドを実行して、クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. 次のコマンドを実行して、AWS 上に OpenID Connect ID プロバイダーと S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1. **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
3. **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

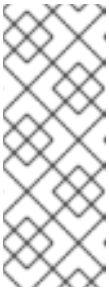
   2 **install-config.yaml** ファイルの場所を指定します。

   3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   c. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

   ```
   $ ccoctl aws create-iam-roles \
     --name=<name> \
     --region=<aws_region> \
   ```

```
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```



注記

GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.5.5.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

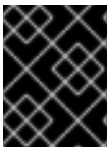
   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

6.3.5.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。

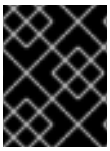
注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.5.7. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.5.8. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

6.3.5.9. 次のステップ

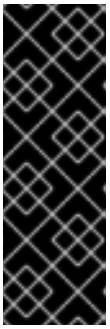
- インストールを検証 します。
- クラスターをカスタマイズ します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

6.3.6. AWS のクラスターの既存 VPC へのインストール

OpenShift Container Platform バージョン 4.16 では、Amazon Web Services (AWS) 上の既存の Amazon Virtual Private Cloud (VPC) にクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

6.3.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。
- 既存の VPC がクラスターとは異なるアカウントによって所有されている場合は、アカウント間で VPC を共有 しています。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

6.3.6.2. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS) の既存の Amazon Virtual Private Cloud (VPC) における既存サブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

6.3.6.2.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- クラスターが使用するアベイラビリティーゾーンごとにパブリックサブネットとプライベートサブネットを作成します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。このタイプの設定の例は、AWS ドキュメントの パブリックサブネットとプライベートサブネット (NAT) を使用した VPC を参照してください。
各サブネット ID を記録します。インストールを完了するには、 **install-config.yaml** ファイルの **プラットフォーム** セクションにこれらの値を入力する必要があります。AWS ドキュメントの サブネット ID の検索 を参照してください。
- VPC の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれている必要があります。サブネット CIDR ブロックは、指定したマシン CIDR に属している必要があります。
- VPC には、パブリックインターネットゲートウェイが接続されている必要があります。アベイラビリティーゾーンごとに以下が必要です。
  - パブリックサブネットには、インターネットゲートウェイへのルートが必要です。
  - パブリックサブネットには、EIP アドレスが割り当てられた NAT ゲートウェイが必要です。
  - プライベートサブネットには、パブリックサブネットの NAT ゲートウェイへのルートが必要です。
- VPC は **kubernetes.io/cluster/.*: owned**、**Name**、**openshift.io/cluster** タグを使用できません。
インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。**Name** タグは EC2 **Name** フィールドと重複し、その結果インストールが失敗するため、使用できません。
- OpenShift Container Platform クラスターを AWS Outpost に拡張し、既存の Outpost サブネットを使用する場合、既存のサブネットで **kubernetes.io/cluster/unmanaged: true** タグを使用する必要があります。このタグを適用しないと、Cloud Controller Manager が Outpost サブネットにサービスロードバランサーを作成するため、インストールが失敗する可能性があります。これはサポートされていない設定です。
- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドと **platform.aws.hostedZoneRole** フィールドを使用して、ホストゾーンを定義できます。クラスターをインストールするアカウントとプライベート

ホストゾーンを共有することで、別のアカウントからプライベートホストゾーンを使用できます。別のアカウントからプライベートホストゾーンを使用する場合は、**Passthrough** または **Manual** 認証情報モードを使用する必要があります。

非接続環境で作業している場合、EC2、ELB、および S3 エンドポイントのパブリック IP アドレスに到達することはできません。インストール中にインターネットトラフィックを制限するレベルに応じて、次の設定オプションを使用できます。

オプション 1: VPC エンドポイントを作成する
VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

このオプションを使用すると、VPC および必要な AWS サービスの間でネットワークトラフィックがプライベートのままになります。

オプション 2: VPC エンドポイントなしでプロキシーを作成する
インストールプロセスの一環として、HTTP または HTTPS プロキシーを設定できます。このオプションを使用すると、インターネットトラフィックはプロキシーを経由して、必要な AWS サービスに到達します。

オプション 3: VPC エンドポイントでプロキシーを作成する
インストールプロセスの一環として、VPC エンドポイントを使用して HTTP または HTTPS プロキシーを設定できます。VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

install-config.yaml ファイルでプロキシーを設定するときに、これらのエンドポイントを **noProxy** フィールドに追加します。このオプションを使用すると、プロキシーはクラスターがインターネットに直接アクセスするのを防ぎます。ただし、VPC と必要な AWS サービスの間のネットワークトラフィックはプライベートのままです。

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。

6.3.6.2.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

6.3.6.2.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

6.3.6.2.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

6.3.6.2.5. オプション: AWS セキュリティーグループ

デフォルトでは、インストールプログラムは、セキュリティーグループを作成し、コントロールプレーンとコンピュートマシンに接続します。デフォルトのセキュリティーグループに関連付けられたルールは変更できません。

ただし、既存の VPC に関連付けられている追加の既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用できます。カスタムセキュリティーグループを適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

インストールプロセスの一環として、クラスターをデプロイする前に **install-config.yaml** ファイルを変更してカスタムセキュリティーグループを適用します。

詳細は、「既存の AWS セキュリティーグループのクラスターへの適用」を参照してください。

6.3.6.2.6. 共有 VPC にインストールする場合の信頼ポリシーの変更

共有 VPC を使用してクラスターをインストールする場合は、**Passthrough** または **Manual** 認証情報モードを使用できます。クラスターをインストールするために使用される IAM ロールを、VPC を所有するアカウントの信頼ポリシーのプリンシパルとして追加する必要があります。

Passthrough モードを使用する場合は、クラスターを作成するアカウントの Amazon Resource Name (ARN) (**arn:aws:iam::123456789012:user/clustercreator** など) をプリンシパルとして信頼ポリシーに追加します。

Manual モードを使用する場合は、クラスターを作成するアカウントの ARN と、クラスター所有者アカウントの Ingress オペレーターロールの ARN (**arn:aws:iam::123456789012:role/<cluster-name>-openshift-ingress-operator-cloud-credentials** など) をプリンシパルとして信頼ポリシーに追加します。

次のアクションをポリシーに追加する必要があります。

> **例6.28 共有 VPC のインストールに必要なアクション**
>
> - **route53:ChangeResourceRecordSets**
> - **route53:ListHostedZones**
> - **route53:ListHostedZonesByName**
> - **route53:ListResourceRecordSets**

- **route53:ChangeTagsForResource**
- **route53:GetAccountLimit**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**
- **tag:GetResources**
- **tag:UntagResources**

6.3.6.3. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

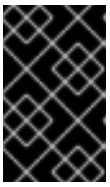
iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- AWS のインストール設定パラメーター

6.3.6.3.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

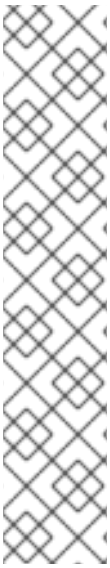
表6.15 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

6.3.6.3.2. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services(AWS)インスタンスタイプは OpenShift Container Platform でテストされています。

注記

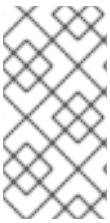
AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

例6.29 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

6.3.6.3.3. 64 ビット ARM インフラストラクチャー上の AWS のテスト済みインスタンスタイプ

次の Amazon Web Services (AWS) 64 ビット ARM インスタンスタイプは、OpenShift Container Platform でテストされています。



注記

AWS ARM インスタンスには、次のチャートに含まれるマシンタイプを使用してください。チャートに記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、クラスターインストールの最小リソース要件に記載されている最小リソース要件と一致していることを確認してください。

例6.30 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

6.3.6.3.4. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

インストール設定ファイル **install-config.yaml** をカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      metadataService:
        authentication: Optional (7)
      type: m6i.xlarge
  replicas: 3
compute: (8)
- hyperthreading: Enabled (9)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (10)
      metadataService:
        authentication: Optional (11)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (12)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (13)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (14)
```

```
    propagateUserTags: true 15
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 16
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-0c5d3e03c0ab9b19a 17
    serviceEndpoints: 18
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 19
fips: false 20
sshKey: ssh-ed25519 AAAA... 21
pullSecret: '{"auths": ...}' 22
```

1 12 14 22 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 8 15 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

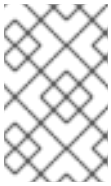
5 9 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 10 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

7 11 Amazon EC2 Instance Metadata Service v2 (IMDSv2) を要求するかどうか。IMDSv2 を要求するには、パラメーター値を **Required** に設定します。IMDSv1 と IMDSv2 の両方の使用を許可するには、パラメーター値を **Optional** に設定します。値が指定されていない場合、IMDSv1 と IMDSv2 の両方が許可されます。



注記

クラスターのインストール中に設定されるコントロールプレーンマシンの IMDS 設定は、AWS CLI を使用してのみ変更できます。コンピュートマシンの IMDS 設定は、コンピュートマシンセットを使用して変更できます。

13 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

16 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

17 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

18 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

19 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

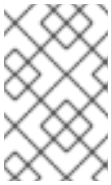
20 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

21 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

6.3.6.3.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキ

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。
4.

指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

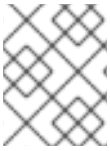
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.3.6.3.6. 既存の AWS セキュリティーグループをクラスターに適用する

既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

前提条件

- AWS でセキュリティーグループを作成している。詳細は、セキュリティーグループ の操作に関する AWS ドキュメントを参照してください。
- セキュリティーグループは、クラスターをデプロイする既存の VPC に関連付ける必要があります。セキュリティーグループを別の VPC に関連付けることはできません。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** ファイルで、**compute.platform.aws.additionalSecurityGroupIDs** パラメーターを編集して、コンピュートマシンに 1 つ以上のカスタムセキュリティーグループを指定します。
2. **controlPlane.platform.aws.AdditionalSecurityGroupIDs** パラメーターを編集して、コントロールプレーンマシンに 1 つ以上のカスタムセキュリティーグループを指定します。
3. ファイルを保存し、クラスターをデプロイする際に参照します。

カスタムセキュリティーグループを指定するサンプル install-config.yaml ファイル

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
    subnets: (2)
      - subnet-1
      - subnet-2
      - subnet-3
```

(1) Amazon EC2 コンソールに表示されるセキュリティーグループの名前を、**sg** 接頭辞を含めて指定します。

(2) クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

6.3.6.4. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように AWS クラスターを設定する の手順に従ってください。

6.3.6.4.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

   サンプル CredentialsRequest オブジェクト

   -

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

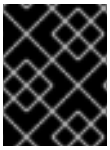
シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
```

```
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

6.3.6.4.2. 短期認証情報を使用するように AWS クラスターを設定

AWS Security Token Service (STS) を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な AWS リソースを作成する必要があります。

6.3.6.4.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

例6.31 必要な AWS パーミッション

必要な iam 権限

- **iam:CreateOpenIDConnectProvider**
- **iam:CreateRole**
- **iam:DeleteOpenIDConnectProvider**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetOpenIDConnectProvider**
- **iam:GetRole**
- **iam:GetUser**
- **iam:ListOpenIDConnectProviders**
- **iam:ListRolePolicies**

- **iam:ListRoles**
- **iam:PutRolePolicy**
- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

必要な s3 権限

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

必要な cloudfront 権限

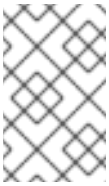
- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

例6.32 CloudFront を使用したプライベート S3 バケットに対する追加の権限

- **cloudfront:CreateCloudFrontOriginAccessIdentity**

- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**
- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**



注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

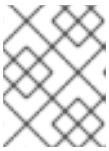
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。

- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

6.3.6.4.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

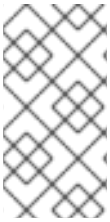
AWS リソースを作成するときは、次のオプションがあります。

- **ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。これはリソースを作成する最も簡単な方法です。単一コマンドでの AWS リソースの作成 を参照してください。
- AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。AWS リソースの個別の作成 を参照してください。

6.3.6.4.2.2.1. 単一コマンドでの AWS リソースの作成

ccoctl ツールが AWS リソースの作成に使用するプロセスが組織の要件を自動的に満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。詳細は、「AWS リソースの個別の作成」を参照してください。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

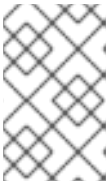
注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

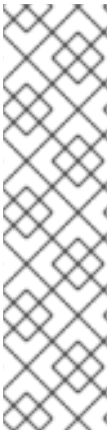
```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.6.4.2.2.2. AWS リソースの個別の作成

ccoctl ツールを使用して、AWS リソースを個別に作成できます。このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。詳細は、「単一コマンドによる AWS リソースの作成」を参照してください。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. 次のコマンドを実行して、クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. 次のコマンドを実行して、AWS 上に OpenID Connect ID プロバイダーと S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \1
  --region=<aws_region> \2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1. **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
3. **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   c. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

   ```
   $ ccoctl aws create-iam-roles \
     --name=<name> \
     --region=<aws_region> \
   ```

```
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

注記

GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.6.4.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

6.3.6.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。

注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

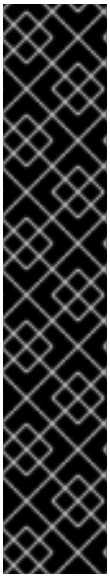
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.6.6. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.6.7. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

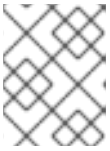
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

6.3.6.8. 次のステップ

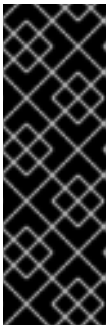
- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。
- AWS 上の既存の VPC にクラスターをインストールした後、AWS VPC クラスターを AWS Outpost に拡張 できます。

6.3.7. プライベートクラスターの AWS へのインストール

OpenShift Container Platform バージョン 4.16 では、Amazon Web Services (AWS) 上の既存の VPC にプライベートクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

6.3.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

6.3.7.2. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。
- 以下にアクセスできるマシンからデプロイ。
    - プロビジョニングするクラウドの API サービス。

- プロビジョニングするネットワーク上のホスト。
- インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

6.3.7.2.1. AWS のプライベートクラスター

Amazon Web Services (AWS) でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、プライベートネットワークからのみアクセスできるように Ingress Operator および API サーバーを設定します。

クラスターには、引き続き AWS API にアクセスするためにインターネットへのアクセスが必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックロードバランサー
- クラスターの **baseDomain** に一致するパブリック Route 53 ゾーン

インストールプログラムは、プライベート Route 53 ゾーンを作成するために指定する **baseDomain** とクラスターに必要なレコードを使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

6.3.7.2.1.1. 制限事項

プライベートクラスターにパブリック機能を追加する機能には制限があります。

- Kubernetes API エンドポイントは、追加のアクションを実行せずにインストールする場合はパブリックにすることができません。これらのアクションには、使用中のアベイラビリティーゾーンごとに VPC でパブリックサブネットやパブリックのロードバランサーを作成することや、6443 のインターネットからのトラフィックを許可するようにコントロールプレーンのセキュリティーグループを設定することなどが含まれます。
- パブリックのサービスタイプのロードバランサーを使用する場合には、各アベイラビリティーゾーンのパブリックサブネットに **kubernetes.io/cluster/<cluster-infra-id>: shared** のタグを付け、AWS がそれらを使用してパブリックロードバランサーを作成できるようにします。

6.3.7.3. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS) の既存の Amazon Virtual Private Cloud (VPC) における既存サブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ロードバ

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

6.3.7.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- VPC は **kubernetes.io/cluster/.*: owned**、**Name**、**openshift.io/cluster** タグを使用できません。
  インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。**Name** タグは EC2 **Name** フィールドと重複し、その結果インストールが失敗するため、使用できません。
- OpenShift Container Platform クラスターを AWS Outpost に拡張し、既存の Outpost サブネットを使用する場合、既存のサブネットで **kubernetes.io/cluster/unmanaged: true** タグを

使用する必要があります。このタグを適用しないと、Cloud Controller Manager が Outpost サブネットにサービスロードバランサーを作成するため、インストールが失敗する可能性があります。これはサポートされていない設定です。

- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドと **platform.aws.hostedZoneRole** フィールドを使用して、ホストゾーンを定義できます。クラスターをインストールするアカウントとプライベートホストゾーンを共有することで、別のアカウントからプライベートホストゾーンを使用できます。別のアカウントからプライベートホストゾーンを使用する場合は、**Passthrough** または **Manual** 認証情報モードを使用する必要があります。

非接続環境で作業している場合、EC2、ELB、および S3 エンドポイントのパブリック IP アドレスに到達することはできません。インストール中にインターネットトラフィックを制限するレベルに応じて、次の設定オプションを使用できます。

オプション 1: VPC エンドポイントを作成する
VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

このオプションを使用すると、VPC および必要な AWS サービスの間でネットワークトラフィックがプライベートのままになります。

オプション 2: VPC エンドポイントなしでプロキシーを作成する
インストールプロセスの一環として、HTTP または HTTPS プロキシーを設定できます。このオプションを使用すると、インターネットトラフィックはプロキシーを経由して、必要な AWS サービスに到達します。

オプション 3: VPC エンドポイントでプロキシーを作成する
インストールプロセスの一環として、VPC エンドポイントを使用して HTTP または HTTPS プロキシーを設定できます。VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

install-config.yaml ファイルでプロキシーを設定するときに、これらのエンドポイントを **noProxy** フィールドに追加します。このオプションを使用すると、プロキシーはクラスターがインターネットに直接アクセスするのを防ぎます。ただし、VPC と必要な AWS サービスの間のネットワークトラフィックはプライベートのままです。

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。</td></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明	
		0 - 65535	アウトバウンド一時 (ephemeral) トラフィック
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。	

6.3.7.3.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

6.3.7.3.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネン

トの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

6.3.7.3.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

6.3.7.3.5. オプション: AWS セキュリティーグループ

デフォルトでは、インストールプログラムは、セキュリティーグループを作成し、コントロールプレーンとコンピュートマシンに接続します。デフォルトのセキュリティーグループに関連付けられたルールは変更できません。

ただし、既存の VPC に関連付けられている追加の既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用できます。カスタムセキュリティーグループを適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

インストールプロセスの一環として、クラスターをデプロイする前に **install-config.yaml** ファイルを変更してカスタムセキュリティーグループを適用します。

詳細は、「既存の AWS セキュリティーグループのクラスターへの適用」を参照してください。

6.3.7.4. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

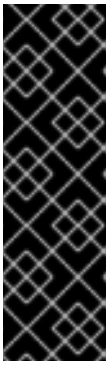
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

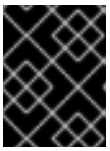
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- AWS のインストール設定パラメーター

6.3.7.4.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表6.16 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

6.3.7.4.2. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services(AWS)インスタンスタイプは OpenShift Container Platform でテストされています。

注記

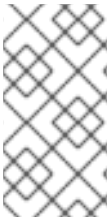
AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

> **例6.33 64 ビット x86 アーキテクチャーに基づくマシンタイプ**
>
> - **c4.***
> - **c5.***
> - **c5a.***

- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

6.3.7.4.3. 64 ビット ARM インフラストラクチャー上の AWS のテスト済みインスタンスタイプ

次の Amazon Web Services (AWS) 64 ビット ARM インスタンスタイプは、OpenShift Container Platform でテストされています。



注記

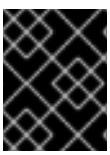
AWS ARM インスタンスには、次のチャートに含まれるマシンタイプを使用してください。チャートに記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、クラスターインストールの最小リソース要件に記載されている最小リソース要件と一致していることを確認してください。

例6.34 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

6.3.7.4.4. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

インストール設定ファイル **install-config.yaml** をカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      metadataService:
        authentication: Optional (7)
      type: m6i.xlarge
  replicas: 3
compute: (8)
- hyperthreading: Enabled (9)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (10)
      metadataService:
        authentication: Optional (11)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (12)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (13)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (14)
    propagateUserTags: true (15)
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: (16)
    - subnet-1
```

```
    - subnet-2
    - subnet-3
    amiID: ami-0c5d3e03c0ab9b19a 17
    serviceEndpoints: 18
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 19
fips: false 20
sshKey: ssh-ed25519 AAAA... 21
publish: Internal 22
pullSecret: '{"auths": ...}' 23
```

1 12 14 23 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 8 15 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 9 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 10 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

7 11 Amazon EC2 Instance Metadata Service v2 (IMDSv2) を要求するかどうか。IMDSv2 を要求するには、パラメーター値を **Required** に設定します。IMDSv1 と IMDSv2 の両方の使用を許可するには、パラメーター値を **Optional** に設定します。値が指定されていない場合、IMDSv1 と IMDSv2 の両方が許可されます。

注記

クラスターのインストール中に設定されるコントロールプレーンマシンの IMDS 設定は、AWS CLI を使用してのみ変更できます。コンピュートマシンの IMDS 設定は、コンピュートマシンセットを使用して変更できます。

13

インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

16 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

17 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

18 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

19 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

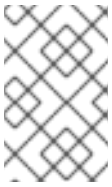
20 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

21 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

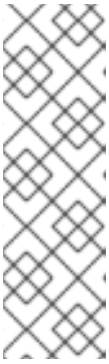
22 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

6.3.7.4.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの

trustedCA フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

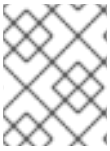
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.3.7.4.6. 既存の AWS セキュリティーグループをクラスターに適用する

既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

前提条件

- AWS でセキュリティーグループを作成している。詳細は、セキュリティーグループ の操作に関する AWS ドキュメントを参照してください。
- セキュリティーグループは、クラスターをデプロイする既存の VPC に関連付ける必要があります。セキュリティーグループを別の VPC に関連付けることはできません。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** ファイルで、**compute.platform.aws.additionalSecurityGroupIDs** パラメーターを編集して、コンピュートマシンに 1 つ以上のカスタムセキュリティーグループを指定します。

2. **controlPlane.platform.aws.AdditionalSecurityGroupIDs** パラメーターを編集して、コントロールプレーンマシンに 1 つ以上のカスタムセキュリティーグループを指定します。

3. ファイルを保存し、クラスターをデプロイする際に参照します。

カスタムセキュリティーグループを指定するサンプル install-config.yaml ファイル

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
    subnets: (2)
      - subnet-1
      - subnet-2
      - subnet-3
```

(1) Amazon EC2 コンソールに表示されるセキュリティーグループの名前を、**sg** 接頭辞を含めて指定します。

(2) クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

6.3.7.5. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。

- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように AWS クラスターを設定する の手順に従ってください。

6.3.7.5.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

   サンプル CredentialsRequest オブジェクト

   -

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
```

```
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

重要

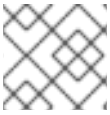
手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

6.3.7.5.2. 短期認証情報を使用するように AWS クラスターを設定

AWS Security Token Service (STS) を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な AWS リソースを作成する必要があります。

6.3.7.5.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

例6.35 必要な AWS パーミッション

必要な iam 権限

- **iam:CreateOpenIDConnectProvider**
- **iam:CreateRole**
- **iam:DeleteOpenIDConnectProvider**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetOpenIDConnectProvider**
- **iam:GetRole**
- **iam:GetUser**
- **iam:ListOpenIDConnectProviders**
- **iam:ListRolePolicies**

- **iam:ListRoles**
- **iam:PutRolePolicy**
- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

必要な s3 権限

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

必要な cloudfront 権限

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

例6.36 CloudFront を使用したプライベート S3 バケットに対する追加の権限

- **cloudfront:CreateCloudFrontOriginAccessIdentity**

- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**
- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。

- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

6.3.7.5.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

AWS リソースを作成するときは、次のオプションがあります。

- **ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。これはリソースを作成する最も簡単な方法です。単一コマンドでの AWS リソースの作成 を参照してください。
- AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。AWS リソースの個別の作成 を参照してください。

6.3.7.5.2.2.1. 単一コマンドでの AWS リソースの作成

ccoctl ツールが AWS リソースの作成に使用するプロセスが組織の要件を自動的に満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。詳細は、「AWS リソースの個別の作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

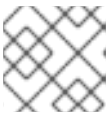
1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。



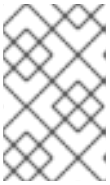
注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.7.5.2.2.2. AWS リソースの個別の作成

ccoctl ツールを使用して、AWS リソースを個別に作成できます。このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。詳細は、「単一コマンドによる AWS リソースの作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. 次のコマンドを実行して、クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. 次のコマンドを実行して、AWS 上に OpenID Connect ID プロバイダーと S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \1
  --region=<aws_region> \2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1. **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
3. **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

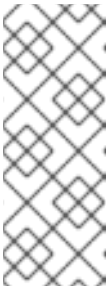
   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   c. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

   ```
   $ ccoctl aws create-iam-roles \
     --name=<name> \
     --region=<aws_region> \
   ```

```
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```



注記

GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.7.5.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

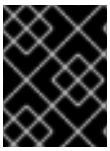
   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

6.3.7.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

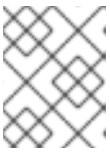
1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。



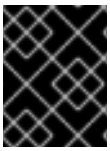
注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

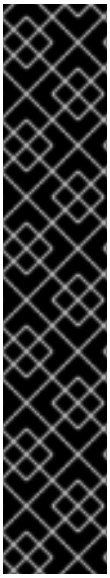


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.7.7. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.7.8. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

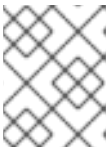
```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

6.3.7.9. 次のステップ

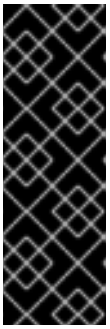
- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

6.3.8. AWS の government リージョンへのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、Amazon Web Services (AWS) 上のクラスターを government リージョンにインストールできます。リージョンを設定するには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

6.3.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

6.3.8.2. AWS government リージョン

OpenShift Container Platform は、AWS Gov Cloud (US) リージョンへのクラスターのデプロイをサポートします。

以下の AWS GovCloud パーティションがサポートされます。

- **us-gov-east-1**
- **us-gov-west-1**

6.3.8.3. インストール要件

クラスターをインストールする前に、以下を行う必要があります。

- クラスターをホストするために、既存のプライベート AWS VPC とサブネットを提供します。パブリックゾーンは、AWS GovCloud の Route 53 ではサポートされません。その結果、AWS government リージョンにデプロイする場合、クラスターはプライベートである必要があります。
- インストール設定ファイル (**install-config.yaml**) を手動で作成します。

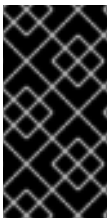
6.3.8.4. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

注記

パブリックゾーンは、AWS GovCloud リージョンの Route 53 ではサポートされていません。したがって、クラスターを AWS GovCloud リージョンにデプロイする場合は、クラスターをプライベートにする必要があります。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。



重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。
- 以下にアクセスできるマシンからデプロイ。
  - プロビジョニングするクラウドの API サービス。
  - プロビジョニングするネットワーク上のホスト。
  - インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

6.3.8.4.1. AWS のプライベートクラスター

Amazon Web Services (AWS) でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、プライベートネットワークからのみアクセスできるように Ingress Operator および API サーバーを設定します。

クラスターには、引き続き AWS API にアクセスするためにインターネットへのアクセスが必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックロードバランサー
- クラスターの **baseDomain** に一致するパブリック Route 53 ゾーン

インストールプログラムは、プライベート Route 53 ゾーンを作成するために指定する **baseDomain** とクラスターに必要なレコードを使用します。クラスターは、Operator がクラスターのパブリックレ

コードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

6.3.8.4.1.1. 制限事項

プライベートクラスターにパブリック機能を追加する機能には制限があります。

- Kubernetes API エンドポイントは、追加のアクションを実行せずにインストールする場合はパブリックにすることができません。これらのアクションには、使用中のアベイラビリティーゾーンごとに VPC でパブリックサブネットやパブリックのロードバランサーを作成することや、6443 のインターネットからのトラフィックを許可するようにコントロールプレーンのセキュリティーグループを設定することなどが含まれます。
- パブリックのサービスタイプのロードバランサーを使用する場合には、各アベイラビリティーゾーンのパブリックサブネットに **kubernetes.io/cluster/<cluster-infra-id>: shared** のタグを付け、AWS がそれらを使用してパブリックロードバランサーを作成できるようにします。

6.3.8.5. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS) の既存の Amazon Virtual Private Cloud (VPC) における既存サブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

6.3.8.5.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC を使用するインストールプログラムおよびクラス

カスタム VPC を使用する場合は、その カスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- VPC は **kubernetes.io/cluster/.*: owned**、**Name**、**openshift.io/cluster** タグを使用できません。
  インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。**Name** タグは EC2 **Name** フィールドと重複し、その結果インストールが失敗するため、使用できません。

- OpenShift Container Platform クラスターを AWS Outpost に拡張し、既存の Outpost サブネットを使用する場合、既存のサブネットで **kubernetes.io/cluster/unmanaged: true** タグを使用する必要があります。このタグを適用しないと、Cloud Controller Manager が Outpost サブネットにサービスロードバランサーを作成するため、インストールが失敗する可能性があります。これはサポートされていない設定です。

- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドと **platform.aws.hostedZoneRole** フィールドを使用して、ホストゾーンを定義できます。クラスターをインストールするアカウントとプライベートホストゾーンを共有することで、別のアカウントからプライベートホストゾーンを使用できます。別のアカウントからプライベートホストゾーンを使用する場合は、**Passthrough** または **Manual** 認証情報モードを使用する必要があります。

非接続環境で作業している場合、EC2、ELB、および S3 エンドポイントのパブリック IP アドレスに到達することはできません。インストール中にインターネットトラフィックを制限するレベルに応じて、次の設定オプションを使用できます。

オプション 1: VPC エンドポイントを作成する
VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

このオプションを使用すると、VPC および必要な AWS サービスの間でネットワークトラフィックがプライベートのままになります。

オプション 2: VPC エンドポイントなしでプロキシーを作成する
インストールプロセスの一環として、HTTP または HTTPS プロキシーを設定できます。このオプションを使用すると、インターネットトラフィックはプロキシーを経由して、必要な AWS サービスに到達します。

オプション 3: VPC エンドポイントでプロキシーを作成する
インストールプロセスの一環として、VPC エンドポイントを使用して HTTP または HTTPS プロキシーを設定できます。VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

install-config.yaml ファイルでプロキシーを設定するときに、これらのエンドポイントを **noProxy** フィールドに追加します。このオプションを使用すると、プロキシーはクラスターがインターネットに直接アクセスするのを防ぎます。ただし、VPC と必要な AWS サービスの間のネットワークトラフィックはプライベートのままです。

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。
パブリックサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::SubnetNetworkAclAssociation**	VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
<tr><td>プライベートサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::RouteTable
AWS::EC2::SubnetRouteTableAssociation</td><td>VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。</td></tr>
</table>

6.3.8.5.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

6.3.8.5.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

6.3.8.5.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

6.3.8.5.5. オプション: AWS セキュリティーグループ

デフォルトでは、インストールプログラムは、セキュリティーグループを作成し、コントロールプレーンとコンピュートマシンに接続します。デフォルトのセキュリティーグループに関連付けられたルールは変更できません。

ただし、既存の VPC に関連付けられている追加の既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用できます。カスタムセキュリティーグループを適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

インストールプロセスの一環として、クラスターをデプロイする前に **install-config.yaml** ファイルを変更してカスタムセキュリティーグループを適用します。

詳細は、「既存の AWS セキュリティーグループのクラスターへの適用」を参照してください。

6.3.8.6. AWS Marketplace イメージの取得

AWS Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に AWS を通じてサブスクライブする必要があります。オファーにサブスクライブすると、インストールプログラムがコンピュートノードのデプロイに使用する AMI ID が提供されます。

前提条件

- オファーを購入するための AWS アカウントを持っている。このアカウントは、クラスターのインストールに使用されるアカウントと同じである必要はありません。

手順

1. AWS Marketplace で OpenShift Container Platform サブスクリプションを完了します。
2. ご使用の AWS リージョンの AMI ID を記録します。インストールプロセスの一環として、クラスターをデプロイする前に、この値で **install-config.yaml** ファイルを更新する必要があります。

AWS Marketplace コンピュートノードを含む install-config.yaml ファイルのサンプル

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

1. AWS Marketplace サブスクリプションの AMI ID。
2. AMI ID は特定の AWS リージョンに関連付けられています。インストール設定ファイルを作成するときは、サブスクリプションの設定時に指定したものと同じ AWS リージョンを選択してください。

6.3.8.7. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

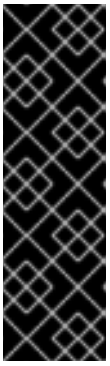
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- AWS のインストール設定パラメーター

6.3.8.7.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表6.17 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

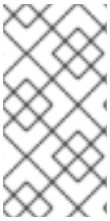
プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

6.3.8.7.2. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services(AWS)インスタンスタイプは OpenShift Container Platform でテストされています。



注記

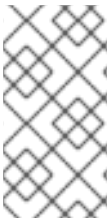
AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

例6.37 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

6.3.8.7.3. 64 ビット ARM インフラストラクチャー上の AWS のテスト済みインスタンスタイプ

次の Amazon Web Services (AWS) 64 ビット ARM インスタンスタイプは、OpenShift Container Platform でテストされています。



注記

AWS ARM インスタンスには、次のチャートに含まれるマシンタイプを使用してください。チャートに記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、クラスターインストールの最小リソース要件に記載されている最小リソース要件と一致していることを確認してください。

例6.38 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

6.3.8.7.4. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

インストール設定ファイル **install-config.yaml** をカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。これを使用して、手動で作成したインストール設定ファイルにパラメーター値を入力します。

```
apiVersion: v1
baseDomain: example.com 1
credentialsMode: Mint 2
controlPlane: 3 4
  hyperthreading: Enabled 5
  name: master
  platform:
    aws:
      zones:
      - us-gov-west-1a
      - us-gov-west-1b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 6
      metadataService:
        authentication: Optional 7
      type: m6i.xlarge
  replicas: 3
compute: 8
- hyperthreading: Enabled 9
  name: worker
  platform:
    aws:
      rootVolume:
```

```
        iops: 2000
        size: 500
        type: io1 10
      metadataService:
        authentication: Optional 11
      type: c5.4xlarge
      zones:
      - us-gov-west-1c
  replicas: 3
metadata:
  name: test-cluster 12
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 13
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-gov-west-1 14
    propagateUserTags: true 15
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 16
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-0c5d3e03c0ab9b19a 17
    serviceEndpoints: 18
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 19
fips: false 20
sshKey: ssh-ed25519 AAAA... 21
publish: Internal 22
pullSecret: '{"auths": ...}' 23
```

1 12 14 23 必須。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 8 15 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1

つのコントロールプレーンプールのみが使用されます。

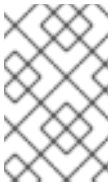
5 9 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 10 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

7 11 Amazon EC2 Instance Metadata Service v2 (IMDSv2) を要求するかどうか。IMDSv2 を要求するには、パラメーター値を **Required** に設定します。IMDSv1 と IMDSv2 の両方の使用を許可するには、パラメーター値を **Optional** に設定します。値が指定されていない場合、IMDSv1 と IMDSv2 の両方が許可されます。



注記

クラスターのインストール中に設定されるコントロールプレーンマシンの IMDS 設定は、AWS CLI を使用してのみ変更できます。コンピュートマシンの IMDS 設定は、コンピュートマシンセットを使用して変更できます。

13 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

16 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

17 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

18 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

19 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

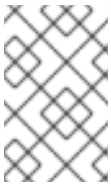
20 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

21 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

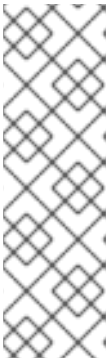
22 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

6.3.8.7.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。

4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

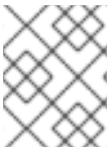
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.3.8.7.6. 既存の AWS セキュリティーグループをクラスターに適用する

既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

前提条件

- AWS でセキュリティーグループを作成している。詳細は、セキュリティーグループ の操作に関する AWS ドキュメントを参照してください。
- セキュリティーグループは、クラスターをデプロイする既存の VPC に関連付ける必要があります。セキュリティーグループを別の VPC に関連付けることはできません。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** ファイルで、**compute.platform.aws.additionalSecurityGroupIDs** パラメーターを編集して、コンピュートマシンに 1 つ以上のカスタムセキュリティーグループを指定します。
2. **controlPlane.platform.aws.AdditionalSecurityGroupIDs** パラメーターを編集して、コントロールプレーンマシンに 1 つ以上のカスタムセキュリティーグループを指定します。
3. ファイルを保存し、クラスターをデプロイする際に参照します。

カスタムセキュリティーグループを指定するサンプル install-config.yaml ファイル

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
```

```
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
    subnets: (2)
      - subnet-1
      - subnet-2
      - subnet-3
```

(1) Amazon EC2 コンソールに表示されるセキュリティーグループの名前を、**sg** 接頭辞を含めて指定します。

(2) クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

6.3.8.8. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- 個々のコンポーネントのクラスター外部で管理される短期認証情報を実装するには、Cloud Credential Operator ユーティリティーマニフェストの組み込み に記載された手順に従います。

6.3.8.8.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   ```

```
# ...
```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
```

```
    - iam:ListAccessKeys
    resource: "*"
  ...
```

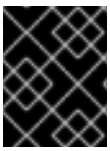
5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```



重要

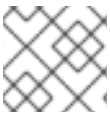
手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

6.3.8.8.2. 短期認証情報を使用するように AWS クラスターを設定

AWS Security Token Service (STS) を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な AWS リソースを作成する必要があります。

6.3.8.8.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

  例6.39 必要な AWS パーミッション

  必要な iam 権限

  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**
  - **iam:TagOpenIDConnectProvider**
  - **iam:TagRole**

  必要な s3 権限

  - **s3:CreateBucket**
  - **s3:DeleteBucket**
  - **s3:DeleteObject**

- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

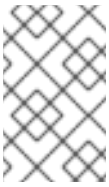
必要な cloudfront 権限

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

例6.40 CloudFront を使用したプライベート S3 バケットに対する追加の権限

- **cloudfront:CreateCloudFrontOriginAccessIdentity**
- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**
- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**



注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

6.3.8.8.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

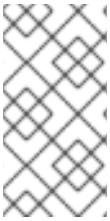
AWS リソースを作成するときは、次のオプションがあります。

- **ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。これはリソースを作成する最も簡単な方法です。単一コマンドでの AWS リソースの作成 を参照してください。
- AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。AWS リソースの個別の作成 を参照してください。

6.3.8.8.2.2.1. 単一コマンドでの AWS リソースの作成

ccoctl ツールが AWS リソースの作成に使用するプロセスが組織の要件を自動的に満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。詳細は、「AWS リソースの個別の作成」を参照してください。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバ

イダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.8.8.2.2.2. AWS リソースの個別の作成

ccoctl ツールを使用して、AWS リソースを個別に作成できます。このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。詳細は、「単一コマンドによる AWS リソースの作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. 次のコマンドを実行して、クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. 次のコマンドを実行して、AWS 上に OpenID Connect ID プロバイダーと S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1. **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
3. **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at .well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

   2 **install-config.yaml** ファイルの場所を指定します。

   3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   c. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

   ```
   $ ccoctl aws create-iam-roles \
     --name=<name> \
     --region=<aws_region> \
     --credentials-requests-dir=<path_to_credentials_requests_directory> \
     --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
   <aws_region>.amazonaws.com
   ```

   注記

   GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

   クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

   それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container

Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.8.8.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

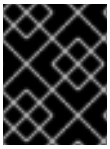
```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

6.3.8.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

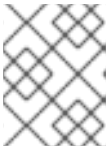
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。



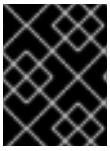
注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

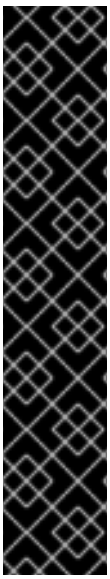


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.8.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.8.11. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

6.3.8.12. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

6.3.9. AWS 上のクラスターをシークレットまたはトップシークレットリージョンにインストールする

OpenShift Container Platform バージョン 4.16 では、Amazon Web Services (AWS) 上のクラスターを以下のシークレットリージョンにインストールできます。

- シークレット Commercial Cloud Services (SC2S)
- Commercial Cloud Services (C2S)

いずれかのリージョンでクラスターを設定するには、クラスターをインストールする前に、**install config.yaml** ファイルのパラメーターを変更します。

警告

OpenShift Container Platform 4.16 では、インストールプログラムは Terraform の代わりに Cluster API を使用して、AWS へのインストール中にクラスターインフラストラクチャーをプロビジョニングします。Cluster API 実装を使用した AWS のクラスターをシークレットまたはトップシークレットリージョンにインストールすることは、OpenShift Container Platform 4.16 のリリース時点ではテストされていません。このドキュメントは、シークレットリージョンへのインストールがテストされると更新されます。

シークレットリージョンまたは上位シークレットリージョンのセキュリティーグループに対するネットワークロードバランサーのサポートには、これらのリージョンでのインストールが失敗するという既知の問題があります。詳細は、OCPBUGS-31080 を参照してください。

6.3.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。

重要

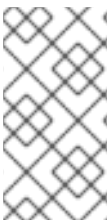
コンピューターに AWS プロファイルが保存されている場合は、多要素認証デバイスの使用中に生成した一時的なセッショントークンを使用しないでください。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

6.3.9.2. AWS シークレットリージョン

次の AWS シークレットパーティションがサポートされています。

- **us-isob-east-1** (SC2S)
- **us-iso-east-1** (C2S)



注記

AWS SC2S および C2S リージョンでサポートされる最大 MTU は、AWS コマーシャルと同じではありません。インストール中の MTU の設定の詳細については、**ネットワークをカスタマイズした AWS へのクラスターのインストール** の **クラスターネットワークオペレーターの設定オブジェクト** セクションを参照してください。

6.3.9.3. インストール要件

Red Hat は、AWS Secret およびトップシークレットリージョン用の Red Hat Enterprise Linux CoreOS (RHCOS) Amzaon Machine Image を公開していません。

クラスターをインストールする前に、以下を行う必要があります。

- カスタム RHCOS AMI をアップロードします。
- インストール設定ファイル (**install-config.yaml**) を手動で作成します。
- インストール設定ファイルで、AWS リージョンおよび付随するカスタム AMI を指定します。

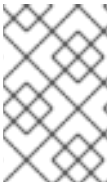
OpenShift Container Platform インストールプログラムを使用してインストール設定ファイルを作成することはできません。インストーラーは RHCOS AMI のネイティブサポートのない AWS リージョンをリスト表示しません。

重要

AWS API にはカスタム CA 信頼バンドルが必要なため、**install-config.yaml** ファイルの **additionalTrustBundle** フィールドで、カスタム CA 証明書も定義する必要があります。インストールプログラムが AWS API にアクセスできるようにするには、インストールプログラムを実行するマシンに CA 証明書を定義する必要もあります。マシン上のトラストストアに CA バンドルを追加するか、**AWS_CA_BUNDLE** 環境変数を使用するか、AWS 設定ファイルの **ca_bundle** フィールドで CA バンドルを定義する必要があります。

6.3.9.4. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。



注記

パブリックゾーンは、AWS トップシークレットリージョンの Route 53 ではサポートされていません。したがって、クラスターを AWS トップシークレットリージョンにデプロイする場合は、クラスターをプライベートにする必要があります。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。
- 以下にアクセスできるマシンからデプロイ。
  - プロビジョニングするクラウドの API サービス。
  - プロビジョニングするネットワーク上のホスト。
  - インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

6.3.9.4.1. AWS のプライベートクラスター

Amazon Web Services (AWS) でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、プライベートネットワークからのみアクセスできるように Ingress Operator および API サーバーを設定します。

クラスターには、引き続き AWS API にアクセスするためにインターネットへのアクセスが必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックロードバランサー
- クラスターの **baseDomain** に一致するパブリック Route 53 ゾーン

インストールプログラムは、プライベート Route 53 ゾーンを作成するために指定する **baseDomain** とクラスターに必要なレコードを使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

6.3.9.4.1.1. 制限事項

プライベートクラスターにパブリック機能を追加する機能には制限があります。

- Kubernetes API エンドポイントは、追加のアクションを実行せずにインストールする場合はパブリックにすることができません。これらのアクションには、使用中のアベイラビリティーゾーンごとに VPC でパブリックサブネットやパブリックのロードバランサーを作成することや、6443 のインターネットからのトラフィックを許可するようにコントロールプレーンのセキュリティーグループを設定することなどが含まれます。
- パブリックのサービスタイプのロードバランサーを使用する場合には、各アベイラビリティーゾーンのパブリックサブネットに **kubernetes.io/cluster/<cluster-infra-id>: shared** のタグを付け、AWS がそれらを使用してパブリックロードバランサーを作成できるようにします。

6.3.9.5. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS) の既存の Amazon Virtual Private Cloud (VPC) における既存サブネットにクラスターをデプロイできます。OpenShift Container

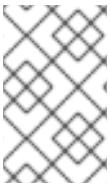
Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

6.3.9.5.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- VPC は **kubernetes.io/cluster/.*: owned**、**Name**、**openshift.io/cluster** タグを使用できません。
  インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サ

ブネットに追加できるようにします。**Name** タグは EC2 **Name** フィールドと重複し、その結果インストールが失敗するため、使用できません。

- OpenShift Container Platform クラスターを AWS Outpost に拡張し、既存の Outpost サブネットを使用する場合、既存のサブネットで **kubernetes.io/cluster/unmanaged: true** タグを使用する必要があります。このタグを適用しないと、Cloud Controller Manager が Outpost サブネットにサービスロードバランサーを作成するため、インストールが失敗する可能性があります。これはサポートされていない設定です。

- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドと **platform.aws.hostedZoneRole** フィールドを使用して、ホストゾーンを定義できます。クラスターをインストールするアカウントとプライベートホストゾーンを共有することで、別のアカウントからプライベートホストゾーンを使用できます。別のアカウントからプライベートホストゾーンを使用する場合は、**Passthrough** または **Manual** 認証情報モードを使用する必要があります。

SC2S または C2S リージョンのクラスターは、EC2、ELB、および S3 エンドポイントのパブリック IP アドレスに到達できません。インストール中にインターネットトラフィックを制限するレベルに応じて、次の設定オプションを使用できます。

オプション 1: VPC エンドポイントを作成する
VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

SC2S

- **elasticloadbalancing.<aws_region>.sc2s.sgov.gov**
- **ec2.<aws_region>.sc2s.sgov.gov**
- **s3.<aws_region>.sc2s.sgov.gov**

C2S

- **elasticloadbalancing.<aws_region>.c2s.ic.gov**
- **ec2.<aws_region>.c2s.ic.gov**
- **s3.<aws_region>.c2s.ic.gov**

このオプションを使用すると、VPC および必要な AWS サービスの間でネットワークトラフィックがプライベートのままになります。

オプション 2: VPC エンドポイントなしでプロキシーを作成する
インストールプロセスの一環として、HTTP または HTTPS プロキシーを設定できます。このオプションを使用すると、インターネットトラフィックはプロキシーを経由して、必要な AWS サービスに到達します。

オプション 3: VPC エンドポイントでプロキシーを作成する

インストールプロセスの一環として、VPC エンドポイントを使用して HTTP または HTTPS プロキシーを設定できます。VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

SC2S

- **elasticloadbalancing.<aws_region>.sc2s.sgov.gov**
- **ec2.<aws_region>.sc2s.sgov.gov**
- **s3.<aws_region>.sc2s.sgov.gov**

C2S

- **elasticloadbalancing.<aws_region>.c2s.ic.gov**
- **ec2.<aws_region>.c2s.ic.gov**
- **s3.<aws_region>.c2s.ic.gov**

install-config.yaml ファイルでプロキシーを設定するときに、これらのエンドポイントを **noProxy** フィールドに追加します。このオプションを使用すると、プロキシーはクラスターがインターネットに直接アクセスするのを防ぎます。ただし、VPC と必要な AWS サービスの間のネットワークトラフィックはプライベートのままです。

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。
パブリックサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::SubnetNetworkAclAssociation**	VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
<tr><td>プライベートサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::RouteTable
AWS::EC2::SubnetRouteTableAssociation</td><td>VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。</td></tr>
</table>

6.3.9.5.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

6.3.9.5.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

6.3.9.5.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

6.3.9.5.5. オプション: AWS セキュリティーグループ

デフォルトでは、インストールプログラムは、セキュリティーグループを作成し、コントロールプレーンとコンピュートマシンに接続します。デフォルトのセキュリティーグループに関連付けられたルールは変更できません。

ただし、既存の VPC に関連付けられている追加の既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用できます。カスタムセキュリティーグループを適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

インストールプロセスの一環として、クラスターをデプロイする前に **install-config.yaml** ファイルを変更してカスタムセキュリティーグループを適用します。

詳細は、「既存の AWS セキュリティーグループのクラスターへの適用」を参照してください。

6.3.9.6. AWS でのカスタム RHCOS AMI のアップロード

カスタム Amazon Web Services (AWS) リージョンにデプロイする場合、そのリージョンに属するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) をアップロードする必要があります。

前提条件

- AWS アカウントを設定している。
- 必要な IAM サービスロール で、Amazon S3 バケットを作成している。
- RHCOS VMDK ファイルを Amazon S3 にアップロードしている。RHCOS VMDK ファイルは、インストールする OpenShift Container Platform のバージョンと同じか、それ以下のバージョンである必要があります。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer を参照してください。

手順

1. AWS プロファイルを環境変数としてエクスポートします。

```
$ export AWS_PROFILE=<aws_profile> 1
```

2. カスタム AMI に関連付けるリージョンを環境変数としてエクスポートします。

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

3. 環境変数として Amazon S3 にアップロードした RHCOS のバージョンをエクスポートします。

```
$ export RHCOS_VERSION=<version> 1
```

1 1 1 **4.16.0** などの RHCOS VMDK バージョン。

4. Amazon S3 バケット名を環境変数としてエクスポートします。

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. **containers.json** ファイルを作成し、RHCOS VMDK ファイルを定義します。

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
  }
}
EOF
```

6. RHCOS ディスクを Amazon EBS スナップショットとしてインポートします。

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1 **rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64** などの RHCOS ディスクがインポートされていることの説明。

2 RHCOS ディスクを説明する JSON ファイルへのファイルパス。JSON ファイルには、Amazon S3 バケット名とキーが含まれている必要があります。

7. イメージインポートのステータスを確認します。

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

出力例

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

SnapshotId をコピーして、イメージを登録します。

8. RHCOS スナップショットからカスタム RHCOS AMI を作成します。

```
$ aws ec2 register-image \
  --region ${AWS_DEFAULT_REGION} \
  --architecture x86_64 \ 1
  --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
  --ena-support \
  --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
  --virtualization-type hvm \
  --root-device-name '/dev/xvda' \
  --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1. **x86_64**、**aarch64**、**s390x**、または **ppc64le** などの RHCOS VMDK アーキテクチャータイプ。
2. インポートされたスナップショットの **Description**。
3. RHCOS AMI の名前。
4. インポートされたスナップショットからの **SnapshotID**。

これらの API の詳細は、AWS ドキュメントの Importing a Disk as a Snapshot Using VM Import/Export および Creating a Linux AMI from a snapshot を参照してください。

6.3.9.7. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

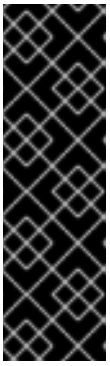
前提条件

- カスタムの RHCOS AMI をアップロードしている。
- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

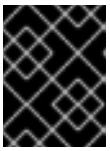
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- AWS のインストール設定パラメーター

6.3.9.7.1. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services(AWS)インスタンスタイプは OpenShift Container Platform でテストされています。

注記

AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

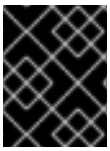
例6.41 シークレット領域の 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **i3.***
- **m4.***
- **m5.***

- **r4.***
- **r5.***
- **t3.***

6.3.9.7.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

インストール設定ファイル **install-config.yaml** をカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。これを使用して、手動で作成したインストール設定ファイルにパラメーター値を入力します。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-iso-east-1a
      - us-iso-east-1b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      metadataService:
        authentication: Optional (7)
      type: m6i.xlarge
  replicas: 3
compute: (8)
- hyperthreading: Enabled (9)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (10)
      metadataService:
        authentication: Optional (11)
      type: c5.4xlarge
      zones:
      - us-iso-east-1a
      - us-iso-east-1b
  replicas: 3
```

```
metadata:
  name: test-cluster (12)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (13)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-iso-east-1 (14)
    propagateUserTags: true (15)
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: (16)
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 (17) (18)
    serviceEndpoints: (19)
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV (20)
fips: false (21)
sshKey: ssh-ed25519 AAAA... (22)
publish: Internal (23)
pullSecret: '{"auths": ...}' (24)
additionalTrustBundle: | (25)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
```

(1) (12) (14) (17) (24) 必須。

(2) オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

(3) (8) (15) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(4) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(5) (9) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチ

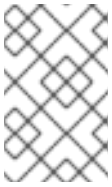
スレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 10 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

7 11 Amazon EC2 Instance Metadata Service v2 (IMDSv2) を要求するかどうか。IMDSv2 を要求するには、パラメーター値を **Required** に設定します。IMDSv1 と IMDSv2 の両方の使用を許可するには、パラメーター値を **Optional** に設定します。値が指定されていない場合、IMDSv1 と IMDSv2 の両方が許可されます。



注記

クラスターのインストール中に設定されるコントロールプレーンマシンの IMDS 設定は、AWS CLI を使用してのみ変更できます。コンピュートマシンの IMDS 設定は、コンピュートマシンセットを使用して変更できます。

13 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

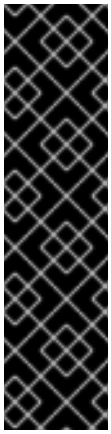
16 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

18 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

19 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

20 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

21 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

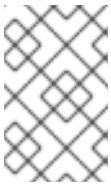


重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

22 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

23 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

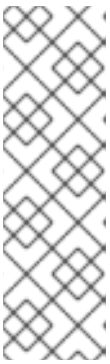
25 カスタム CA 証明書。これは、SC2S または C2S リージョンにデプロイするときに必要です。これは、AWS API がカスタム CA 信頼バンドルを必要とするためです。

6.3.9.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> (1)
  httpsProxy: https://<username>:<pswd>@<ip>:<port> (2)
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com (3)
additionalTrustBundle: | (4)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> (5)
```

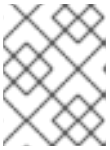
(1) クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。

(2) クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

(3) プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。

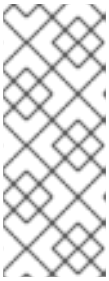
(4) 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

(5) オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.3.9.7.4. 既存の AWS セキュリティーグループをクラスターに適用する

既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

前提条件

- AWS でセキュリティーグループを作成している。詳細は、セキュリティーグループ の操作に関する AWS ドキュメントを参照してください。
- セキュリティーグループは、クラスターをデプロイする既存の VPC に関連付ける必要があります。セキュリティーグループを別の VPC に関連付けることはできません。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** ファイルで、**compute.platform.aws.additionalSecurityGroupIDs** パラメーターを編集して、コンピュートマシンに 1 つ以上のカスタムセキュリティーグループを指定します。
2. **controlPlane.platform.aws.AdditionalSecurityGroupIDs** パラメーターを編集して、コントロールプレーンマシンに 1 つ以上のカスタムセキュリティーグループを指定します。
3. ファイルを保存し、クラスターをデプロイする際に参照します。

カスタムセキュリティーグループを指定するサンプル install-config.yaml ファイル

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
```

```
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
    subnets: (2)
      - subnet-1
      - subnet-2
      - subnet-3
```

(1) Amazon EC2 コンソールに表示されるセキュリティーグループの名前を、**sg** 接頭辞を含めて指定します。

(2) クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

6.3.9.8. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように AWS クラスターを設定する の手順に従ってください。

6.3.9.8.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   ```

```
# ...
```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
```

```
      - iam:ListAccessKeys
      resource: "*"
    ...
```

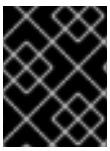
5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

6.3.9.8.2. 短期認証情報を使用するように AWS クラスターを設定

AWS Security Token Service (STS) を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な AWS リソースを作成する必要があります。

6.3.9.8.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

  例6.42 必要な AWS パーミッション

  必要な iam 権限

  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**
  - **iam:TagOpenIDConnectProvider**
  - **iam:TagRole**

  必要な s3 権限

  - **s3:CreateBucket**
  - **s3:DeleteBucket**
  - **s3:DeleteObject**

- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

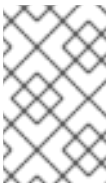
必要な cloudfront 権限

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

例6.43 CloudFront を使用したプライベート S3 バケットに対する追加の権限

- **cloudfront:CreateCloudFrontOriginAccessIdentity**
- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**
- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**



注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

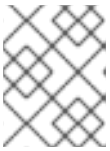
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

6.3.9.8.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

AWS リソースを作成するときは、次のオプションがあります。

- **ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。これはリソースを作成する最も簡単な方法です。単一コマンドでの AWS リソースの作成 を参照してください。
- AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。AWS リソースの個別の作成 を参照してください。

6.3.9.8.2.2.1. 単一コマンドでの AWS リソースの作成

ccoctl ツールが AWS リソースの作成に使用するプロセスが組織の要件を自動的に満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。詳細は、「AWS リソースの個別の作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバ

イダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

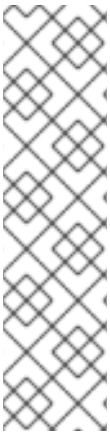
```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.9.8.2.2.2. AWS リソースの個別の作成

ccoctl ツールを使用して、AWS リソースを個別に作成できます。このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。詳細は、「単一コマンドによる AWS リソースの作成」を参照してください。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. 次のコマンドを実行して、クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. 次のコマンドを実行して、AWS 上に OpenID Connect ID プロバイダーと S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \1
  --region=<aws_region> \2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1 **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。

2 **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。

3 **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at .well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

   2 **install-config.yaml** ファイルの場所を指定します。

   3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   c. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

   ```
   $ ccoctl aws create-iam-roles \
     --name=<name> \
     --region=<aws_region> \
     --credentials-requests-dir=<path_to_credentials_requests_directory> \
     --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
   <aws_region>.amazonaws.com
   ```

   注記

   GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

   クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

   それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container

Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.9.8.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

6.3.9.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。

注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.9.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.9.11. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

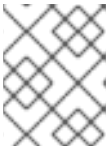
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

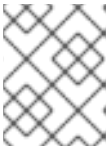


注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- Web コンソールへのアクセス

6.3.9.12. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

6.3.10. AWS China でのクラスターのアンインストール

OpenShift Container Platform バージョン 4.16 では、クラスターを以下の Amazon Web Services (AWS) China リージョンにインストールできます。

- **cn-north-1** (Beijing)
- **cn-northwest-1** (Ningxia)

6.3.10.1. 前提条件

- インターネットコンテンツプロバイダー (ICP) ライセンスがある。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

6.3.10.2. インストール要件

Red Hat は、AWS China リージョンの Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) を公開しません。

クラスターをインストールする前に、以下を行う必要があります。

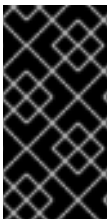
- カスタム RHCOS AMI をアップロードします。
- インストール設定ファイル (**install-config.yaml**) を手動で作成します。
- インストール設定ファイルで、AWS リージョンおよび付随するカスタム AMI を指定します。

OpenShift Container Platform インストールプログラムを使用してインストール設定ファイルを作成することはできません。インストーラーは RHCOS AMI のネイティブサポートのない AWS リージョンをリスト表示しません。

6.3.10.3. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。



重要

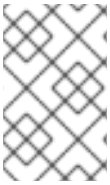
クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。
- 以下にアクセスできるマシンからデプロイ。
  - プロビジョニングするクラウドの API サービス。
  - プロビジョニングするネットワーク上のホスト。

- インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホストを使用できます。



注記

AWS China は、VPC とネットワーク間の VPN 接続をサポートしません。Beijing および Ningxia リージョンの Amazon VPC サービスの詳細は、AWS China ドキュメントの Amazon Virtual Private Cloud を参照してください。

6.3.10.3.1. AWS のプライベートクラスター

Amazon Web Services (AWS) でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、プライベートネットワークからのみアクセスできるように Ingress Operator および API サーバーを設定します。

クラスターには、引き続き AWS API にアクセスするためにインターネットへのアクセスが必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックロードバランサー
- クラスターの **baseDomain** に一致するパブリック Route 53 ゾーン

インストールプログラムは、プライベート Route 53 ゾーンを作成するために指定する **baseDomain** とクラスターに必要なレコードを使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

6.3.10.3.1.1. 制限事項

プライベートクラスターにパブリック機能を追加する機能には制限があります。

- Kubernetes API エンドポイントは、追加のアクションを実行せずにインストールする場合はパブリックにすることができません。これらのアクションには、使用中のアベイラビリティーゾーンごとに VPC でパブリックサブネットやパブリックのロードバランサーを作成することや、6443 のインターネットからのトラフィックを許可するようにコントロールプレーンのセキュリティーグループを設定することなどが含まれます。
- パブリックのサービスタイプのロードバランサーを使用する場合には、各アベイラビリティーゾーンのパブリックサブネットに **kubernetes.io/cluster/<cluster-infra-id>: shared** のタグを付け、AWS がそれらを使用してパブリックロードバランサーを作成できるようにします。

6.3.10.4. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS) の既存の Amazon Virtual Private Cloud (VPC) における既存サブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラ

インによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

6.3.10.4.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- VPC は **kubernetes.io/cluster/.*: owned**、**Name**、**openshift.io/cluster** タグを使用できません。
  インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サ

ブネットに追加できるようにします。**Name** タグは EC2 **Name** フィールドと重複し、その結果インストールが失敗するため、使用できません。

- OpenShift Container Platform クラスターを AWS Outpost に拡張し、既存の Outpost サブネットを使用する場合、既存のサブネットで **kubernetes.io/cluster/unmanaged: true** タグを使用する必要があります。このタグを適用しないと、Cloud Controller Manager が Outpost サブネットにサービスロードバランサーを作成するため、インストールが失敗する可能性があります。これはサポートされていない設定です。

- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドと **platform.aws.hostedZoneRole** フィールドを使用して、ホストゾーンを定義できます。クラスターをインストールするアカウントとプライベートホストゾーンを共有することで、別のアカウントからプライベートホストゾーンを使用できます。別のアカウントからプライベートホストゾーンを使用する場合は、**Passthrough** または **Manual** 認証情報モードを使用する必要があります。

非接続環境で作業している場合、EC2、ELB、および S3 エンドポイントのパブリック IP アドレスに到達することはできません。インストール中にインターネットトラフィックを制限するレベルに応じて、次の設定オプションを使用できます。

オプション 1: VPC エンドポイントを作成する
VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com.cn**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

このオプションを使用すると、VPC および必要な AWS サービスの間でネットワークトラフィックがプライベートのままになります。

オプション 2: VPC エンドポイントなしでプロキシーを作成する
インストールプロセスの一環として、HTTP または HTTPS プロキシーを設定できます。このオプションを使用すると、インターネットトラフィックはプロキシーを経由して、必要な AWS サービスに到達します。

オプション 3: VPC エンドポイントでプロキシーを作成する
インストールプロセスの一環として、VPC エンドポイントを使用して HTTP または HTTPS プロキシーを設定できます。VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com.cn**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

install-config.yaml ファイルでプロキシーを設定するときに、これらのエンドポイントを **noProxy** フィールドに追加します。このオプションを使用すると、プロキシーはクラスターがインターネットに直接アクセスするのを防ぎます。ただし、VPC と必要な AWS サービスの間のネットワークトラフィッ

クはプライベートのままです。

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。</td></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明	
		1024 - 65535	インバウンド一時 (ephemeral) トラフィック
		0 - 65535	アウトバウンド一時 (ephemeral) トラフィック
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。	

6.3.10.4.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

6.3.10.4.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

6.3.10.4.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

6.3.10.4.5. オプション: AWS セキュリティーグループ

デフォルトでは、インストールプログラムは、セキュリティーグループを作成し、コントロールプレーンとコンピュートマシンに接続します。デフォルトのセキュリティーグループに関連付けられたルールは変更できません。

ただし、既存の VPC に関連付けられている追加の既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用できます。カスタムセキュリティーグループを適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

インストールプロセスの一環として、クラスターをデプロイする前に **install-config.yaml** ファイルを変更してカスタムセキュリティーグループを適用します。

詳細は、「既存の AWS セキュリティーグループのクラスターへの適用」を参照してください。

6.3.10.5. AWS でのカスタム RHCOS AMI のアップロード

カスタム Amazon Web Services (AWS) リージョンにデプロイする場合、そのリージョンに属するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) をアップロードする必要があります。

前提条件

- AWS アカウントを設定している。
- 必要な IAM サービスロール で、Amazon S3 バケットを作成している。
- RHCOS VMDK ファイルを Amazon S3 にアップロードしている。RHCOS VMDK ファイルは、インストールする OpenShift Container Platform のバージョンと同じか、それ以下のバージョンである必要があります。

- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer を参照してください。

手順

1. AWS プロファイルを環境変数としてエクスポートします。

```
$ export AWS_PROFILE=<aws_profile> 1
```

1 **beijingadmin** などの AWS 認証情報を保持する AWS プロファイル名。

2. カスタム AMI に関連付けるリージョンを環境変数としてエクスポートします。

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

1 **cn-north-1** などの AWS リージョン

3. 環境変数として Amazon S3 にアップロードした RHCOS のバージョンをエクスポートします。

```
$ export RHCOS_VERSION=<version> 1
```

1 **4.16.0** などの RHCOS VMDK バージョン。

4. Amazon S3 バケット名を環境変数としてエクスポートします。

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. **containers.json** ファイルを作成し、RHCOS VMDK ファイルを定義します。

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
  }
}
EOF
```

6. RHCOS ディスクを Amazon EBS スナップショットとしてインポートします。

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1 **rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64** などの RHCOS ディスクがインポートされていることの説明。

2 RHCOS ディスクを説明する JSON ファイルへのファイルパス。JSON ファイルには、Amazon S3 バケット名とキーが含まれている必要があります。

7. イメージインポートのステータスを確認します。

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

出力例

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

SnapshotId をコピーして、イメージを登録します。

8. RHCOS スナップショットからカスタム RHCOS AMI を作成します。

```
$ aws ec2 register-image \
  --region ${AWS_DEFAULT_REGION} \
  --architecture x86_64 \ 1
  --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
  --ena-support \
  --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
  --virtualization-type hvm \
  --root-device-name '/dev/xvda' \
  --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1. **x86_64**、**aarch64**、**s390x**、または **ppc64le** などの RHCOS VMDK アーキテクチャータイプ。
2. インポートされたスナップショットの **Description**。
3. RHCOS AMI の名前。
4. インポートされたスナップショットからの **SnapshotID**。

これらの API の詳細は、AWS ドキュメントの Importing a Disk as a Snapshot Using VM Import/Export および Creating a Linux AMI from a snapshot を参照してください。

6.3.10.6. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

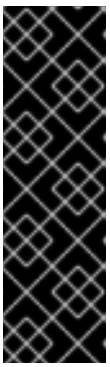
前提条件

- カスタムの RHCOS AMI をアップロードしている。
- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

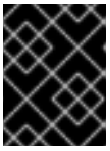
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- AWS のインストール設定パラメーター

6.3.10.6.1. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

インストール設定ファイル **install-config.yaml** をカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。これを使用して、手動で作成したインストール設定ファイルにパラメーター値を入力します。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - cn-north-1a
      - cn-north-1b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      metadataService:
        authentication: Optional (7)
      type: m6i.xlarge
  replicas: 3
compute: (8)
- hyperthreading: Enabled (9)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (10)
      metadataService:
        authentication: Optional (11)
      type: c5.4xlarge
      zones:
      - cn-north-1a
  replicas: 3
metadata:
  name: test-cluster (12)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (13)
  serviceNetwork:
  - 172.30.0.0/16
```

```
platform:
  aws:
    region: cn-north-1 14
    propagateUserTags: true 15
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 16
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 17 18
    serviceEndpoints: 19
      - name: ec2
        url: https://vpce-id.ec2.cn-north-1.vpce.amazonaws.com.cn
    hostedZone: Z3URY6TWQ91KVV 20
fips: false 21
sshKey: ssh-ed25519 AAAA... 22
publish: Internal 23
pullSecret: '{"auths": ...}' 24
```

1 12 14 17 24 必須。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 8 15 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

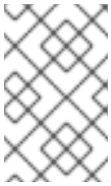
5 9 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 10 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

7 11 Amazon EC2 Instance Metadata Service v2 (IMDSv2) を要求するかどうか。IMDSv2 を要求するに



注記

クラスターのインストール中に設定されるコントロールプレーンマシンの IMDS 設定は、AWS CLI を使用してのみ変更できます。コンピュートマシンの IMDS 設定は、コンピュートマシンセットを使用して変更できます。

13 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

16 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

18 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

19 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

20 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

21 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

22 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

23 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

6.3.10.6.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表6.18 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

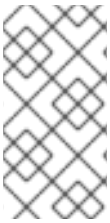
プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

6.3.10.6.3. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services(AWS)インスタンスタイプは OpenShift Container Platform でテストされています。



注記

AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

例6.44 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

6.3.10.6.4. 64 ビット ARM インフラストラクチャー上の AWS のテスト済みインスタンスタイプ

次の Amazon Web Services (AWS) 64 ビット ARM インスタンスタイプは、OpenShift Container Platform でテストされています。

注記

AWS ARM インスタンスには、次のチャートに含まれるマシンタイプを使用してください。チャートに記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、クラスターインストールの最小リソース要件に記載されている最小リソース要件と一致していることを確認してください。

例6.45 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

6.3.10.6.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
```

```
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

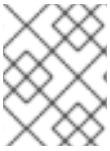
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.3.10.6.6. 既存の AWS セキュリティーグループをクラスターに適用する

既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

前提条件

- AWS でセキュリティーグループを作成している。詳細は、セキュリティーグループ の操作に関する AWS ドキュメントを参照してください。
- セキュリティーグループは、クラスターをデプロイする既存の VPC に関連付ける必要があります。セキュリティーグループを別の VPC に関連付けることはできません。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** ファイルで、**compute.platform.aws.additionalSecurityGroupIDs** パラメーターを編集して、コンピュートマシンに 1 つ以上のカスタムセキュリティーグループを指定します。
2. **controlPlane.platform.aws.AdditionalSecurityGroupIDs** パラメーターを編集して、コントロールプレーンマシンに 1 つ以上のカスタムセキュリティーグループを指定します。
3. ファイルを保存し、クラスターをデプロイする際に参照します。

カスタムセキュリティーグループを指定するサンプル install-config.yaml ファイル

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
```

```
region: us-east-1
subnets: 2
  - subnet-1
  - subnet-2
  - subnet-3
```

1 Amazon EC2 コンソールに表示されるセキュリティーグループの名前を、**sg** 接頭辞を含めて指定します。

2 クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

6.3.10.7. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように AWS クラスターを設定する の手順に従ってください。

6.3.10.7.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
```

```
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

6.3.10.7.2. 短期認証情報を使用するように AWS クラスターを設定

AWS Security Token Service (STS) を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な AWS リソースを作成する必要があります。

6.3.10.7.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。

- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

  例6.46 必要な AWS パーミッション

  必要な iam 権限

  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**
  - **iam:TagOpenIDConnectProvider**
  - **iam:TagRole**

  必要な s3 権限

  - **s3:CreateBucket**
  - **s3:DeleteBucket**
  - **s3:DeleteObject**
  - **s3:GetBucketAcl**
  - **s3:GetBucketTagging**
  - **s3:GetObject**
  - **s3:GetObjectAcl**
  - **s3:GetObjectTagging**
  - **s3:ListBucket**
  - **s3:PutBucketAcl**
  - **s3:PutBucketPolicy**

- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

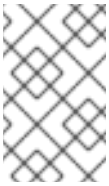
必要な cloudfront 権限

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

例6.47 CloudFront を使用したプライベート S3 バケットに対する追加の権限

- **cloudfront:CreateCloudFrontOriginAccessIdentity**
- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**
- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**



注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

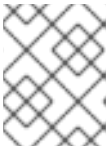
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
```

```
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

6.3.10.7.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

AWS リソースを作成するときは、次のオプションがあります。

- **ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。これはリソースを作成する最も簡単な方法です。単一コマンドでの AWS リソースの作成 を参照してください。
- AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。AWS リソースの個別の作成 を参照してください。

6.3.10.7.2.2.1. 単一コマンドでの AWS リソースの作成

ccoctl ツールが AWS リソースの作成に使用するプロセスが組織の要件を自動的に満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。詳細は、「AWS リソースの個別の作成」を参照してください。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

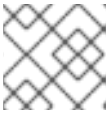
```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
```

```
--included \ 1
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
--to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。



注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

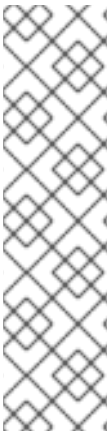
```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.10.7.2.2.2. AWS リソースの個別の作成

ccoctl ツールを使用して、AWS リソースを個別に作成できます。このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。詳細は、「単一コマンドによる AWS リソースの作成」を参照してください。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. 次のコマンドを実行して、クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例

```
2021/04/13 11:01:02 Generating RSA keypair
```

```
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. 次のコマンドを実行して、AWS 上に OpenID Connect ID プロバイダーと S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \1
  --region=<aws_region> \2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1. **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
3. **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at .well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

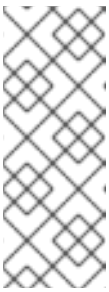
b. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

c. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```



注記

GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールの一覧表示について参照してください。

6.3.10.7.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

6.3.10.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

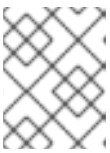
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。



注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.10.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.10.10. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

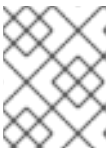
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

6.3.10.11. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

6.3.11. AWS Local Zones 上のコンピュートノードを使用してクラスターをインストールする

install-config.yaml ファイルのエッジコンピュートプールにゾーン名を設定することで、OpenShift Container Platform クラスターを Amazon Web Services (AWS) Local Zones にすばやくインストールできます。または、Local Zone サブネットを持つ既存の Amazon Virtual Private Cloud (VPC) にクラスターをインストールすることもできます。

AWS Local Zones は、クラウドリソースを大都市圏の近くに配置するインフラストラクチャーです。詳細は、AWS Local Zones Documentation を参照してください。

6.3.11.1. インフラストラクチャーの前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択とユーザー用にクラスターを準備する方法 を理解している。
- クラスターをホストするために AWS アカウントを設定 している。

警告

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) を参照してください。
- ファイアウォールを使用している場合は、クラスターがアクセスする必要がある サイトを許可するようにファイアウォールを設定 している。

- ネットワークリソースを作成するために、リージョンとサポートされている AWS ローカルゾーンの場所 を書き留めている。
- AWS ドキュメントの AWS Local Zones features を確認している。
- AWS Local Zones をサポートするネットワークリソースを作成する権限を、アイデンティティーおよびアクセス管理 (IAM) ユーザーまたはロールに追加している。次の例では、AWS Local Zones をサポートするネットワークリソースを作成するためのユーザーまたはロールアクセスを提供できるゾーングループを有効にします。

IAM ユーザーまたはロールに割り当てられた ec2:ModifyAvailabilityZoneGroup 権限を持つ追加の IAM ポリシーの例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:ModifyAvailabilityZoneGroup"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

6.3.11.2. AWS Local Zones とエッジコンピュートプールについて

AWS Local Zones 環境でのインフラストラクチャーの動作とクラスターの制限を理解するには、以降のセクションをお読みください。

6.3.11.2.1. AWS Local Zone でのクラスターの制限

デフォルトのインストール設定で Amazon Web Services (AWS) の Local Zone にクラスターをデプロイする場合、いくつかの制限があります。

重要

次のリストは、事前設定された AWS ゾーンにクラスターをデプロイする場合の制限の詳細を示しています。

- ゾーン内の Amazon EC2 インスタンスとリージョン内の Amazon EC2 インスタンス間の最大伝送単位 (MTU) は **1300** です。これにより、デプロイメントで使用されるネットワークプラグインに応じて、クラスター全体のネットワーク MTU が変わります。
- Network Load Balancer (NLB)、Classic Load Balancer、Network Address Translation (NAT) ゲートウェイなどのネットワークリソースは、グローバルにサポートされていません。
- AWS 上の OpenShift Container Platform クラスターの場合、AWS Elastic Block Storage (EBS) **gp3** タイプのボリュームがノードボリュームのデフォルトであり、ストレージクラスのデフォルトです。このボリュームタイプは、ゾーンの場所ではグローバルに使用できません。デフォルトでは、ゾーン内で実行されるノードは、**gp2** EBS ボリュームを使用してデプロイされます。ゾーンのノードにワークロードを作成する場合は、**gp2-csi StorageClass** パラメーターを設定する必要があります。

インストールプログラムで OpenShift Container Platform クラスターの Local Zone サブネットを自動的に作成する場合、この方法に伴う固有の設定制限が適用されます。

重要

OpenShift Container Platform クラスターのサブネットを自動的に作成するようにインストールプログラムを設定する場合は、次の設定制限が適用されます。

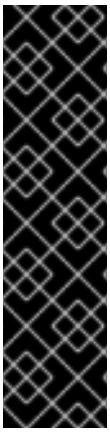
- インストールプログラムは、AWS Local Zones にプライベートサブネットを作成するときに、各サブネットをその親ゾーンのルートテーブルに関連付けます。この操作により、各プライベートサブネットが AWS リージョンの NAT ゲートウェイ経由で Egress トラフィックをインターネットにルーティングできるようになります。
- クラスターのインストール時に親ゾーンのルートテーブルが存在しない場合、インストールプログラムは、プライベートサブネットを Amazon Virtual Private Cloud (VPC) 内の最初に使用可能なプライベートルートテーブルに関連付けます。このアプローチは、OpenShift Container Platform クラスター内の AWS Local Zones サブネットに対してのみ有効です。

6.3.11.2.2. エッジコンピュートプールについて

エッジコンピュートノードは、AWS Local Zones の場所で実行されるテイントされたコンピュートノードです。

Local Zones を使用するクラスターをデプロイする場合は、次の点を考慮してください。

- Local Zone の Amazon EC2 インスタンスは、アベイラビリティゾーンの Amazon EC2 インスタンスよりも高価です。
- AWS Local Zones で実行されているアプリケーションとエンドユーザーの間の遅延は低くなります。たとえば、ローカルゾーンとアベイラビリティゾーンの間で受信トラフィックが混在している場合は、一部のワークロードに遅延の影響が発生します。



重要

通常、Local Zones 内の Amazon EC2 インスタンスとリージョン内の Amazon EC2 インスタンス間の最大伝送単位 (MTU) は 1300 です。クラスターネットワークの MTU は、オーバーヘッドを考慮して、常に EC2 の MTU よりも小さくする必要があります。具体的なオーバーヘッドは、ネットワークプラグインによって決まります。たとえば、OVN-Kubernetes のオーバーヘッドは **100 bytes** です。

ネットワークプラグインは、IPsec などの追加機能を提供できます。MTU のサイズには、このような追加機能も影響します。

詳細は、AWS ドキュメントの ローカルゾーンの仕組み を参照してください。

OpenShift Container Platform 4.12 で、リモートゾーンで使用するために設計された新しいコンピュートプールの **エッジ** が導入されました。エッジコンピュートプール設定は、AWS Local Zone の場所間で共通です。Local Zones リソース上の EC2 や EBS などのリソースのタイプとサイズの制限により、デフォルトのインスタンスタイプが従来のコンピュートプールと異なる場合があります。

Local Zones の場所のデフォルト Elastic Block Store (EBS) は **gp2** であり、非エッジコンピュートプールとは異なります。各 Local Zones に使用される、エッジコンピュートプールのインスタンスタイプも、ゾーンのインスタンスオファリングに応じて、他のコンピュートプールと異なる場合があります。

エッジコンピュートプールは、開発者が AWS Local Zones ノードにアプリケーションをデプロイするために使用できる新しいラベルを作成します。新しいラベルは次のとおりです。

- **node-role.kubernetes.io/edge=''**
- **machine.openshift.io/zone-type=local-zone**
- **machine.openshift.io/zone-group=$ZONE_GROUP_NAME**

デフォルトでは、エッジコンピュートプールのマシンセットは **NoSchedule** テイントを定義して、Local Zones インスタンスに他のワークロードが拡散するのを防ぎます。ユーザーは、Pod 仕様で容認を定義している場合にのみユーザーワークロードを実行できます。

関連情報

- MTU 値の選択
- クラスターネットワークの MTU 変更
- テイントおよび容認 (Toleration) について
- ストレージクラス
- Ingress コントローラーのシャード化

6.3.11.3. インストールの要件

AWS Local Zones 環境にクラスターをインストールする前に、Local Zone 機能を導入できるようにインフラストラクチャーを設定する必要があります。

6.3.11.3.1. AWS Local Zones へのオプトイン

AWS Local Zones にサブネットを作成する予定がある場合は、各ゾーングループに個別にオプトインする必要があります。

前提条件

- AWS CLI をインストールしている。
- OpenShift Container Platform クラスターをデプロイする AWS リージョンを決定しました。
- ゾーングループにオプトインするユーザーまたはロールアカウントに、寛容な IAM ポリシーをアタッチしました。

手順

1. 次のコマンドを実行して、AWS リージョンで利用可能なゾーンをリスト表示します。

   AWS リージョンで利用可能な AWS Local Zones をリスト表示するコマンドの例

   ```
   $ aws --region "<value_of_AWS_Region>" ec2 describe-availability-zones \
       --query 'AvailabilityZones[].[{ZoneName: ZoneName, GroupName: GroupName, Status:
   OptInStatus}]' \
       --filters Name=zone-type,Values=local-zone \
       --all-availability-zones
   ```

   AWS リージョンによっては、利用可能なゾーンのリストが長くなる場合があります。このコマンドは次のフィールドを返します。

   ZoneName

   Local Zones の名前。

   GroupName

   ゾーンを設定するグループ。リージョンにオプトインするには、この名前を保存しておきます。

   Status

   Local Zones グループのステータス。ステータスが **not-opted-in** の場合は、次の手順で説明するように **GroupName** をオプトインする必要があります。

2. 次のコマンドを実行して、AWS アカウントのゾーングループにオプトインします。

   ```
   $ aws ec2 modify-availability-zone-group \
       --group-name "<value_of_GroupName>" \ 1
       --opt-in-status opted-in
   ```

   1 **<value_of_GroupName>** は、サブネットを作成する Local Zones のグループの名前に置き換えます。たとえば、ゾーン **us-east-1-nyc-1a** (米国東部 (ニューヨーク)) を使用するには **us-east-1-nyc-1** と指定します。

6.3.11.3.2. AWS Marketplace イメージの取得

AWS Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に AWS を通じてサブスクライブする必要があります。オファーにサブスクライブすると、インストールプログラムがコンピュートノードのデプロイに使用する AMI ID が提供されます。

前提条件

- オファーを購入するための AWS アカウントを持っている。このアカウントは、クラスターのインストールに使用されるアカウントと同じである必要はありません。

手順

1. AWS Marketplace で OpenShift Container Platform サブスクリプションを完了します。
2. ご使用の AWS リージョンの AMI ID を記録します。インストールプロセスの一環として、クラスターをデプロイする前に、この値で **install-config.yaml** ファイルを更新する必要があります。

AWS Marketplace コンピュートノードを含む install-config.yaml ファイルのサンプル

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) AWS Marketplace サブスクリプションの AMI ID。

(2) AMI ID は特定の AWS リージョンに関連付けられています。インストール設定ファイルを作成するときは、サブスクリプションの設定時に指定したものと同じ AWS リージョンを選択してください。

6.3.11.4. インストールの準備

ノードを Local Zone に拡張する前に、クラスターのインストール環境用に特定のリソースを準備する必要があります。

6.3.11.4.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表6.19 最小リソース要件

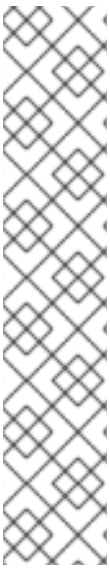
マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

6.3.11.4.2. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services (AWS) インスタンスタイプは、AWS Local Zones で使用するために OpenShift Container Platform でテストされています。

注記

AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

例6.48 AWS Local Zones の 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c5.***
- **c5d.***
- **m6i.***
- **m5.***
- **r5.***
- **t3.***

関連情報

- AWS ドキュメントの AWS Local Zones features を参照してください。

6.3.11.4.3. インストール設定ファイルの作成

インストールプログラムがクラスターをデプロイするために必要なインストール設定ファイルを生成し、カスタマイズします。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャー用の OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- Red Hat が公開している付属の Red Hat Enterprise Linux CoreOS (RHCOS) AMI のある AWS リージョンにクラスターをデプロイしようとしている。カスタム AMI が必要な AWS リージョン (AWS GovCloud リージョンなど) にデプロイする場合は、**install-config.yaml** ファイルを手動で作成する必要があります。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **aws** を選択します。

iii. AWS プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

注記

AWS アクセスキー ID およびシークレットアクセスキーは、インストールホストの現行ユーザーのホームディレクトリーの **~/.aws/credentials** に保存されます。エクスポートされたプロファイルの認証情報がファイルにない場合は、インストールプログラムにより認証情報の入力が求めるプロンプトが出されます。インストールプログラムに指定する認証情報は、ファイルに保存されます。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. オプション: **install-config.yaml** ファイルをバックアップします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

6.3.11.4.4. エッジコンピュートプールを含むインストール設定ファイルの例

次の例は、エッジマシンプール設定を含む **install-config.yaml** ファイルを示しています。

カスタムインスタンスタイプのエッジプールを使用する設定

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
compute:
- name: edge
  platform:
    aws:
      type: r5.2xlarge
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

インスタンスのタイプは場所によって異なります。クラスターを実行する Local Zones で利用可能かどうかを確認するには、AWS のドキュメントを参照してください。

カスタム Amazon Elastic Block Store (EBS) タイプのエッジプールを使用する設定

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
compute:
- name: edge
  platform:
    aws:
      zones:
      - us-west-2-lax-1a
      - us-west-2-lax-1b
      - us-west-2-phx-2a
      rootVolume:
        type: gp3
        size: 120
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

Elastic Block Storage (EBS) タイプは場所によって異なります。クラスターを実行する Local Zones で利用可能かどうかを確認するには、AWS のドキュメントを参照してください。

カスタムセキュリティーグループを持つエッジプールを使用する設定

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
compute:
```

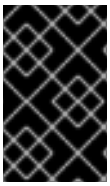
```
- name: edge
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

(1) Amazon EC2 コンソールに表示されるセキュリティーグループの名前を指定します。必ず **sg** 接頭辞を含めてください。

6.3.11.4.5. クラスターネットワークの MTU のカスタマイズ

AWS にクラスターをデプロイする前に、インフラストラクチャーのニーズに合わせて、クラスターネットワークの最大伝送単位 (MTU) をカスタマイズできます。

デフォルトでは、サポートされている Local Zones 機能を使用してクラスターをインストールすると、クラスターネットワークの MTU 値が、ネットワークプラグインによって許可される最低値に自動的に調整されます。



重要

サポートされていない MTU 値を Local Zones インフラストラクチャーで動作する EC2 インスタンスに設定すると、OpenShift Container Platform クラスターに問題が発生する可能性があります。

Local Zone と AWS リージョンの EC2 インスタンス間の MTU 値を引き上げることが可能な場合は、より高い値を手動で設定して、クラスターネットワークのネットワークパフォーマンスを向上できます。

install-config.yaml 設定ファイルで **networking.clusterNetworkMTU** パラメーターを指定することにより、クラスターの MTU をカスタマイズできます。

重要

Local Zones 内のすべてのサブネットは、そのゾーン内の各ノードが AWS リージョン内のサービスと正常に通信してワークロードをデプロイできるように、より高い MTU 値をサポートしている必要があります。

デフォルトの MTU 値を上書きする例

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: edge-zone
networking:
  clusterNetworkMTU: 8901
compute:
- name: edge
  platform:
```

```
    aws:
      zones:
      - us-west-2-lax-1a
      - us-west-2-lax-1b
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

関連情報

- サポートされている最大伝送単位 (MTU) 値の詳細は、AWS ドキュメントの AWS resources supported in Local Zones を参照してください。

6.3.11.5. AWS Local Zones 環境のクラスターインストールオプション

以下のいずれかのインストール方法を選択して、Local Zones に定義されたエッジコンピュートノードを使用して OpenShift Container Platform クラスターを AWS にインストールします。

- 完全に自動化された方法: クラスターをインストールして、コンピュートノードをエッジコンピュートプールに迅速に拡張します。インストールプログラムが、OpenShift Container Platform クラスター用のインフラストラクチャーリソースを自動的に作成します。
- 既存の VPC を使用する方法: AWS 上のクラスターを既存の VPC にインストールします。この場合、Local Zones サブネットを **install-config.yaml** ファイルに指定します。

次のステップ

次のいずれかの方法を選択して、OpenShift Container Platform クラスターを AWS Local Zones 環境にインストールします。

- AWS Local Zones にクラスターを迅速にインストールする
- AWS Local Zone のサブネットが定義されている既存の VPC へのクラスターのインストール

6.3.11.6. AWS Local Zones にクラスターを迅速にインストールする

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS) にクラスターをすばやくインストールして、コンピュートノードを Local Zones の場所に拡張できます。このインストール方法を使用すると、設定ファイルで定義した各ゾーンのネットワークリソースと Local Zones サブネットが、インストールプログラムによって自動的に作成されます。インストールをカスタマイズするには、クラスターをデプロイする前に、**install-config.yaml** ファイル内のパラメーターを変更する必要があります。

6.3.11.6.1. AWS Local Zones を使用するためのインストール設定ファイルの変更

install-config.yaml ファイルを変更して、AWS Local Zones を含めます。

前提条件

- AWS アカウントが設定されている。
- **aws configure** を実行して、AWS キーと AWS リージョンをローカル AWS プロファイルに追加している。

- OpenShift Container Platform クラスターのサブネットを自動的に作成するようにインストールプログラムを指定する際に適用される設定上の制限を理解している。
- 各ゾーンの Local Zones グループにオプトインしている。
- 「インストール設定ファイルの作成」手順を使用して、**install-config.yaml** ファイルを作成している。

手順

1. **install-config.yaml** ファイルを変更して、エッジコンピュートプールの **platform.aws.zones** プロパティーで Local Zones 名を指定します。

```
# ...
platform:
  aws:
    region: <region_name> 1
compute:
- name: edge
  platform:
    aws:
      zones: 2
      - <local_zone_name>
#...
```

1 AWS リージョン名。

2 使用する Local Zones 名のリストは、**platform.aws.region** フィールドで指定した同じ AWS リージョンに存在する必要があります。

エッジノードを Los Angeles と Las Vegas の Local Zones に拡張する us-west-2 AWS リージョンにクラスターをインストールする設定の例

```
apiVersion: v1
baseDomain: example.com
metadata:
  name: cluster-name
platform:
  aws:
    region: us-west-2
compute:
- name: edge
  platform:
    aws:
      zones:
      - us-west-2-lax-1a
      - us-west-2-lax-1b
      - us-west-2-las-1a
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
#...
```

2. クラスターをデプロイします。

関連情報

- インストール設定ファイルの作成
- AWS Local Zone でのクラスターの制限

次のステップ

- クラスターのデプロイ

6.3.11.7. Local Zone のサブネットを持つ既存の VPC へのクラスターのインストール

クラスターを Amazon Web Services (AWS) 上の既存の Amazon Virtual Private Cloud (VPC) にインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

AWS 上のクラスターを既存の VPC にインストールするには、AWS Local Zones を使用してコンピュートノードをクラウドインフラストラクチャーのエッジまで拡張する必要があります。

Local Zone のサブネットは、通常のコンピュートノードをエッジネットワークに拡張します。各エッジコンピュートノードは、ユーザーワークロードを実行します。Amazon Web Services (AWS) の Local Zone 環境を作成し、クラスターをデプロイした後、エッジコンピュートノードを使用して Local Zone のサブネットにユーザーワークロードを作成できます。

注記

プライベートサブネットを作成する場合は、提供されている CloudFormation テンプレートを変更するか、独自のテンプレートを作成する必要があります。

このドキュメントの CloudFormation テンプレートを使用して、ネットワークリソースを作成できます。さらに、テンプレートを変更してインフラストラクチャーをカスタマイズしたり、テンプレートに含まれる情報を使用して会社のポリシーに基づいて AWS リソースを作成したりできます。

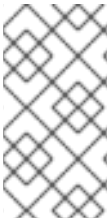
重要

インストーラによってプロビジョニングされたインフラストラクチャーのインストールを実行する手順は、例としてのみ提供されています。既存の VPC にクラスターをインストールするには、クラウドプロバイダーと OpenShift Container Platform のインストールプロセスに関する知識が必要です。CloudFormation テンプレートを使用すると、これらの手順の完了を支援したり、独自のクラスターのインストールをモデル化したりできます。CloudFormation テンプレートを使用してリソースを作成する代わりに、これらのリソースを生成するために他の方法を使用することを決定できます。

6.3.11.7.1. AWS での VPC の作成

OpenShift Container Platform クラスター用の Amazon Web Services (AWS) で、Virtual Private Cloud (VPC) を作成し、Local Zones のすべての場所にサブネットを作成すると、コンピュートノードをエッジロケーションに拡張できます。VPC は、VPN やルートテーブルなどの要件に合わせてさらにカスタマイズできます。初期デプロイメントに含まれていない新しい Local Zones サブネットを追加することもできます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、VPC を表す AWS リソースのスタックを作成できます。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーと AWS リージョンをローカル AWS プロファイルに追加している。
- AWS アカウントで AWS Local Zones にオプトインしている。

手順

1. CloudFormation テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "3" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1. VPC の CIDR ブロック。
2. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
3. VPC をデプロイするアベイラビリティーゾーンの数。
4. **1** から **3** の間の整数を指定します。
5. 各アベイラビリティーゾーン内の各サブネットのサイズ。
6. **5** から **13** の間の整数を指定します。 ここで、**5** は **/27** であり、 **13** は **/19** です。

2. このドキュメントの「VPC の CloudFormation テンプレート」セクションに移動し、記載されているテンプレートから構文をコピーします。コピーしたテンプレートの構文を YAML ファイルとしてローカルシステムに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

3. 次のコマンドを実行して、CloudFormation テンプレートを起動し、VPC を表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> \ 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-vpc** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーターの JSON ファイルの相対パスと名前です。

出力例

```
arn:aws:cloudformation:us-east-1:123456789012:stack/cluster-vpc/dbedae40-2fd3-11eb-820e-12a48460849f
```

4. 次のコマンドを実行して、テンプレートコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーター値を、クラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

VpcId	VPC の ID。
PublicSubnetIds	新規パブリックサブネットの ID。
PrivateSubnetIds	新規プライベートサブネットの ID。
PublicRouteTableId	新しいパブリックルートテーブル ID の ID。

6.3.11.7.2. VPC の CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例6.49 VPC の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
```

```
    EnableDnsHostnames: "true"
    CidrBlock: !Ref VpcCidr
PublicSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 0
    - Fn::GetAZs: !Ref "AWS::Region"
PublicSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 1
    - Fn::GetAZs: !Ref "AWS::Region"
PublicSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
InternetGateway:
  Type: "AWS::EC2::InternetGateway"
GatewayToInternet:
  Type: "AWS::EC2::VPCGatewayAttachment"
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway
PublicRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PublicRoute:
  Type: "AWS::EC2::Route"
  DependsOn: GatewayToInternet
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PublicSubnet2
```

```
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation3:
  Condition: DoAz3
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet3
    RouteTableId: !Ref PublicRouteTable
PrivateSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 0
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
NAT:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP
      - AllocationId
    SubnetId: !Ref PublicSubnet
EIP:
  Type: "AWS::EC2::EIP"
  Properties:
    Domain: vpc
Route:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT
PrivateSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 1
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable2:
```

```
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP2
      - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
```

```
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz3
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
```

```
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
  PrivateRouteTableIds:
    Description: Private Route table IDs
    Value:
      !Join [
        ",",
        [
          !Join ["=", [
            !Select [0, "Fn::GetAZs": !Ref "AWS::Region"],
            !Ref PrivateRouteTable
          ]],
          !If [DoAz2,
              !Join ["=", [!Select [1, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable2]],
              !Ref "AWS::NoValue"
          ],
          !If [DoAz3,
              !Join ["=", [!Select [2, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable3]],
              !Ref "AWS::NoValue"
          ]
        ]
      ]
```

6.3.11.7.3. Local Zones のサブネットの作成

OpenShift Container Platform クラスターのエッジコンピュートノードのマシンセットを設定する前に、Local Zones にサブネットを作成する必要があります。コンピュートノードをデプロイする Local Zone ごとに次の手順を実行してください。

このドキュメントの CloudFormation テンプレートを使用して、CloudFormation スタックを作成できます。その後、このスタックを使用してサブネットをカスタムプロビジョニングできます。

注記

このドキュメントの CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、記載されている情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- Local Zones グループにオプトインしている。

手順

1. このドキュメントの「VPC サブネット用の CloudFormation テンプレート」セクションに移動し、テンプレートから構文をコピーします。コピーしたテンプレートの構文を YAML ファイルとしてローカルシステムに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. 次のコマンドを実行して CloudFormation テンプレートをデプロイします。これにより、VPC を表す AWS リソースのスタックが作成されます。

```
$ aws cloudformation create-stack --stack-name <stack_name> \ 1
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ 2
  --parameters \
    ParameterKey=VpcId,ParameterValue="${VPC_ID}" \ 3
    ParameterKey=ClusterName,ParameterValue="${CLUSTER_NAME}" \ 4
    ParameterKey=ZoneName,ParameterValue="${ZONE_NAME}" \ 5
    ParameterKey=PublicRouteTableId,ParameterValue="${ROUTE_TABLE_PUB}" \ 6
    ParameterKey=PublicSubnetCidr,ParameterValue="${SUBNET_CIDR_PUB}" \ 7
    ParameterKey=PrivateRouteTableId,ParameterValue="${ROUTE_TABLE_PVT}" \ 8
    ParameterKey=PrivateSubnetCidr,ParameterValue="${SUBNET_CIDR_PVT}" 9
```

1. **<stack_name>** は、CloudFormation スタックの名前です (**cluster-wl-<local_zone_shortname>** など)。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルの相対パスと名前です。
3. **${VPC_ID}** は VPC ID であり、VPC 用の CloudFormation テンプレートの出力に含まれる値 **VpcID** です。
4. **${ZONE_NAME}** は、サブネットを作成する Local Zones 名の値です。
5. **${CLUSTER_NAME}** は、新しい AWS リソース名の接頭辞として使用する **ClusterName** の値です。
6. **${SUBNET_CIDR_PUB}** は、パブリックサブネットの作成に使用する有効な CIDR ブロックです。このブロックは、VPC CIDR ブロック **VpcCidr** の一部である必要があります。
7. **${ROUTE_TABLE_PVT}** は、VPC の CloudFormation スタックの出力から抽出した **PrivateRouteTableId** です。
8. **${SUBNET_CIDR_PVT}** は、プライベートサブネットの作成に使用する有効な CIDR ブロックです。このブロックは、VPC CIDR ブロック **VpcCidr** の一部である必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-820e-11eb-2fd3-12a48460849f
```

検証

- 次のコマンドを実行して、テンプレートコンポーネントが存在することを確認します。

  ```
  $ aws cloudformation describe-stacks --stack-name <stack_name>
  ```

  StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーター値を、クラスターを作成するために実行する他の CloudFormation テンプレートに必ず指定してください。

PublicSubnetId	CloudFormation スタックによって作成されたパブリックサブネットの ID。
PrivateSubnetId	CloudFormation スタックによって作成されたプライベートサブネットの ID。

6.3.11.7.4. VPC サブネット用の CloudFormation テンプレート

次の CloudFormation テンプレートを使用して、Local Zones インフラストラクチャー上のゾーンにプライベートサブネットとパブリックサブネットをデプロイできます。

例6.50 VPC サブネット用の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice Subnets (Public and Private)

Parameters:
  VpcId:
    Description: VPC ID that comprises all the target subnets.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster name or prefix name to prepend the Name tag for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.
  ZoneName:
    Description: Zone Name to create the subnets, such as us-west-2-lax-1a.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ZoneName parameter must be specified.
  PublicRouteTableId:
    Description: Public Route Table ID to associate the public subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PublicRouteTableId parameter must be specified.
  PublicSubnetCidr:
```

```
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for public subnet.
    Type: String
  PrivateRouteTableId:
    Description: Private Route Table ID to associate the private subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PrivateRouteTableId parameter must be specified.
  PrivateSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for private subnet.
    Type: String


Resources:
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PublicSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "public", !Ref ZoneName]]

  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTableId

  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PrivateSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "private", !Ref ZoneName]]

  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTableId

Outputs:
  PublicSubnetId:
    Description: Subnet ID of the public subnets.
```

```
    Value:
      !Join ["", [!Ref PublicSubnet]]

  PrivateSubnetId:
    Description: Subnet ID of the private subnets.
    Value:
      !Join ["", [!Ref PrivateSubnet]]
```

関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

6.3.11.7.5. AWS Local Zones サブネットを使用するためのインストール設定ファイルの変更

install-config.yaml ファイルを変更して、Local Zones のサブネットを含めます。

前提条件

- 「Local Zone のサブネットの作成」手順を使用して、サブネットを作成している。
- 「インストール設定ファイルの作成」手順を使用して、**install-config.yaml** ファイルを作成している。

手順

- **install-config.yaml** 設定ファイルを変更して、**platform.aws.subnets** パラメーターで Local Zones のサブネットを指定します。

  Local Zones のサブネットを含むインストール設定ファイルの例

```
# ...
platform:
  aws:
    region: us-west-2
    subnets: 1
    - publicSubnetId-1
    - publicSubnetId-2
    - publicSubnetId-3
    - privateSubnetId-1
    - privateSubnetId-2
    - privateSubnetId-3
    - publicSubnetId-LocalZone-1
# ...
```

1 ゾーン (アベイラビリティーゾーンと Local Zones) 内に作成したサブネット ID のリスト。

関連情報

- 作成した CloudFormation スタックを表示する方法の詳細は、AWS CloudFormation コンソール を参照してください。

- AWS プロファイルと認証情報を設定する方法の詳細は、AWS ドキュメントの Configuration and credential file settings を参照してください。

次のステップ

- クラスターのデプロイ

6.3.11.8. オプション: AWS セキュリティーグループ

デフォルトでは、インストールプログラムは、セキュリティーグループを作成し、コントロールプレーンとコンピュートマシンに接続します。デフォルトのセキュリティーグループに関連付けられたルールは変更できません。

ただし、既存の VPC に関連付けられている追加の既存の AWS セキュリティーグループをコントロールプレーンとコンピュートマシンに適用できます。カスタムセキュリティーグループを適用すると、これらのマシンの受信トラフィックまたは送信トラフィックを制御する必要がある場合に、組織のセキュリティーニーズを満たすことができます。

インストールプロセスの一環として、クラスターをデプロイする前に **install-config.yaml** ファイルを変更してカスタムセキュリティーグループを適用します。

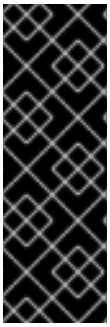
詳細は、「エッジコンピュートプールと AWS Local Zones」を参照してください。

6.3.11.9. オプション: パブリック IP アドレスをエッジコンピュートノードに割り当てます。

ワークロードによっては、Local Zones インフラストラクチャー上のパブリックサブネットにエッジコンピュートノードをデプロイすることが必要になります。その場合は、クラスターのインストール時にマシンセットマニフェストを設定できます。

AWS Local Zones インフラストラクチャーは、指定されたゾーン内のネットワークトラフィックにアクセスします。そのため、そのゾーンに近いエンドユーザーにサービスを提供する際に、アプリケーションで低遅延を利用できます。

デフォルト設定は、プライベートサブネットにコンピュートノードをデプロイする設定であり、お客様のニーズに合わない可能性があります。そのため、インフラストラクチャーにさらにカスタマイズを適用する必要がある場合は、パブリックサブネットにエッジコンピュートノードを作成することを検討してください。



重要

デフォルトでは、OpenShift Container Platform はプライベートサブネットにコンピュートノードをデプロイします。最高のパフォーマンスを得るには、パブリック IP アドレスがサブネットに接続されているサブネットにコンピュートノードを配置することを検討してください。

追加のセキュリティーグループを作成する必要があります。ただし、インターネット経由でグループのルールを開くのは、本当に必要な場合だけに留めてください。

手順

1. インストールプログラムが含まれるディレクトリーに移動し、マニフェストファイルを生成します。インストールマニフェストが **openshift** および **manifests** ディレクトリーレベルに作成されていることを確認します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. インストールプログラムが Local Zones 用に生成するマシンセットマニフェストを編集して、マニフェストがパブリックサブネットにデプロイされるようにします。**spec.template.spec.providerSpec.value.publicIP** パラメーターに **true** を指定します。

クラスターを Local Zones に迅速にインストールするためのマシンセットマニフェスト設定の例

```
spec:
  template:
    spec:
      providerSpec:
        value:
          publicIp: true
          subnet:
            filters:
              - name: tag:Name
                values:
                  - ${INFRA_ID}-public-${ZONE_NAME}
```

Local Zones サブネットを持つ既存の VPC にクラスターをインストールするためのマシンセットマニフェスト設定の例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <infrastructure_id>-edge-<zone>
  namespace: openshift-machine-api
spec:
  template:
    spec:
      providerSpec:
        value:
          publicIp: true
```

6.3.11.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。

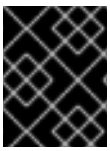
注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.11.11. デプロイしたクラスターのステータスの確認

OpenShift Container Platform が AWS Local Zones に正常にデプロイされたことを確認します。

6.3.11.11.1. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.11.11.2. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

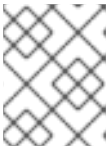
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

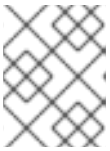


注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- Web コンソールへのアクセス

6.3.11.11.3. エッジコンピューティングプールで作成されたノードの検証

AWS Local Zones インフラストラクチャーを使用するクラスターをインストールしたら、インストール時に作成したマシンセットマニフェストによって作成されたマシンのステータスを確認します。

1. **install-config.yaml** ファイルに追加したサブネットから作成されたマシンセットを確認するには、次のコマンドを実行します。

```
$ oc get machineset -n openshift-machine-api
```

出力例

```
NAME                            DESIRED   CURRENT   READY   AVAILABLE   AGE
cluster-7xw5g-edge-us-east-1-nyc-1a   1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1a       1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1b       1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1c       1         1         1       1           3h4m
```

2. マシンセットから作成されたマシンを確認するには、次のコマンドを実行します。

```
$ oc get machines -n openshift-machine-api
```

出力例

```
NAME                                      PHASE     TYPE        REGION      ZONE            AGE
cluster-7xw5g-edge-us-east-1-nyc-1a-wbclh   Running   c5d.2xlarge   us-east-1   us-east-1-
nyc-1a   3h
cluster-7xw5g-master-0                    Running   m6i.xlarge    us-east-1   us-east-1a      3h4m
cluster-7xw5g-master-1                    Running   m6i.xlarge    us-east-1   us-east-1b      3h4m
cluster-7xw5g-master-2                    Running   m6i.xlarge    us-east-1   us-east-1c      3h4m
cluster-7xw5g-worker-us-east-1a-rtp45     Running   m6i.xlarge    us-east-1   us-east-1a
3h
cluster-7xw5g-worker-us-east-1b-glm7c     Running   m6i.xlarge    us-east-1   us-east-1b
3h
cluster-7xw5g-worker-us-east-1c-qfvz4     Running   m6i.xlarge    us-east-1   us-east-1c
3h
```

3. エッジロールを持つノードを確認するには、次のコマンドを実行します。

```
$ oc get nodes -l node-role.kubernetes.io/edge
```

出力例

```
NAME                          STATUS   ROLES        AGE    VERSION
ip-10-0-207-188.ec2.internal   Ready    edge,worker   172m   v1.25.2+d2e245f
```

次のステップ

- インストールを検証 します。
- 必要に応じて、リモートヘルスをオプトアウト できます。

6.3.12. AWS Wavelength Zones 上のコンピュートノードを使用してクラスターをインストールする

install-config.yaml ファイルのエッジコンピュートプールにゾーン名を設定することで、OpenShift Container Platform クラスターを Amazon Web Services (AWS) Wavelength Zones にすばやくインストールできます。または、Wavelength Zone サブネットを持つ既存の Amazon Virtual Private Cloud (VPC) にクラスターをインストールすることもできます。

AWS Wavelength Zones は、AWS がモバイルエッジコンピューティング (MEC) アプリケーション用に構成したインフラストラクチャーです。

Wavelength Zone は、通信サービスプロバイダー (CSP) の 5G ネットワーク内に AWS のコンピュートサービスとストレージサービスを組み込みます。アプリケーションサーバーを Wavelength Zone に配置することで、5G デバイスからのアプリケーショントラフィックを 5G ネットワーク内に留めることができます。デバイスのアプリケーショントラフィックがターゲットサーバーに直接到達するため、遅延が問題になりません。

関連情報

- AWS ドキュメントの Wavelength Zones を参照してください。

6.3.12.1. インフラストラクチャーの前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択とユーザー用にクラスターを準備する方法 を理解している。
- クラスターをホストするために AWS アカウントを設定 している。

警告

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) を参照してください。
- ファイアウォールを使用している場合は、クラスターがアクセスする必要がある サイトを許可するようにファイアウォールを設定 している。
- ネットワークリソースを作成するために、リージョンとサポートされている AWS Wavelength Zone の場所 を書き留めている。
- AWS ドキュメントの AWS Wavelength features を確認している。
- AWS ドキュメントの Quotas and considerations for Wavelength Zones を確認している。
- AWS Wavelength Zones をサポートするネットワークリソースを作成する権限を、アイデンティティーおよびアクセス管理 (IAM) ユーザーまたはロールに追加している。以下に例を示します。

ユーザーまたはロールに ec2:ModifyAvailabilityZoneGroup、ec2:CreateCarrierGateway、および ec2:DeleteCarrierGateway 権限を割り当てた追加の IAM ポリシーの例

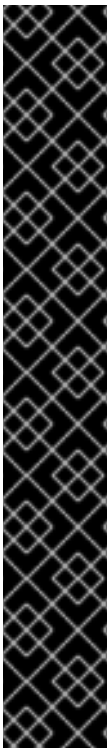
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteCarrierGateway",
        "ec2:CreateCarrierGateway"
      ],
      "Resource": "*"
    },
    {
      "Action": [
        "ec2:ModifyAvailabilityZoneGroup"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

6.3.12.2. AWS Wavelength Zones とエッジコンピュートプールについて

AWS Wavelength Zones 環境でのインフラストラクチャーの動作とクラスターの制限を理解するには、以降のセクションをお読みください。

6.3.12.2.1. AWS Wavelength Zones のクラスターの制限

デフォルトのインストール設定で Amazon Web Services (AWS) の Local Zone にクラスターをデプロイする場合、いくつかの制限があります。



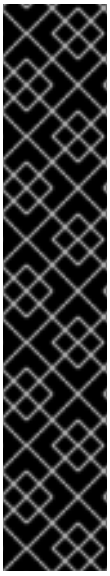
重要

次のリストは、事前設定された AWS ゾーンにクラスターをデプロイする場合の制限の詳細を示しています。

- ゾーン内の Amazon EC2 インスタンスとリージョン内の Amazon EC2 インスタンス間の最大伝送単位 (MTU) は **1300** です。これにより、デプロイメントで使用されるネットワークプラグインに応じて、クラスター全体のネットワーク MTU が変わります。
- Network Load Balancer (NLB)、Classic Load Balancer、Network Address Translation (NAT) ゲートウェイなどのネットワークリソースは、グローバルにサポートされていません。
- AWS 上の OpenShift Container Platform クラスターの場合、AWS Elastic Block Storage (EBS) **gp3** タイプのボリュームがノードボリュームのデフォルトであり、ストレージクラスのデフォルトです。このボリュームタイプは、ゾーンの場所ではグローバルに使用できません。デフォルトでは、ゾーン内で実行されるノードは、**gp2** EBS ボリュームを使用してデプロイされます。ゾーンのノードにワークロードを作成する場合は、**gp2-csi StorageClass** パラメーターを設定する必要があります。

インストールプログラムで OpenShift Container Platform クラスターの Wavelength Zone サブネット

を自動的に作成する場合、この方法に伴う固有の設定制限が適用されます。次の注記で、これらの制限の一部について詳しく説明します。その他の制限については、「インフラストラクチャーの前提条件」セクションに記載されている「Quotas and considerations for Wavelength Zones」ドキュメントをお読みください。



重要

OpenShift Container Platform クラスターのサブネットを自動的に作成するようにインストールプログラムを設定する場合は、次の設定制限が適用されます。

- インストールプログラムは、AWS Wavelength Zones にプライベートサブネットを作成するときに、各サブネットをその親ゾーンのルートテーブルに関連付けます。この操作により、各プライベートサブネットが AWS リージョンの NAT ゲートウェイ経由で Egress トラフィックをインターネットにルーティングできるようになります。
- クラスターのインストール時に親ゾーンのルートテーブルが存在しない場合、インストールプログラムは、プライベートサブネットを Amazon Virtual Private Cloud (VPC) 内の最初に使用可能なプライベートルートテーブルに関連付けます。このアプローチは、OpenShift Container Platform クラスター内の AWS Wavelength Zones サブネットに対してのみ有効です。

6.3.12.2.2. エッジコンピュートプールについて

エッジコンピュートノードは、AWS Wavelength Zones の場所で実行されるテイントされたコンピュートノードです。

Wavelength Zones を使用するクラスターをデプロイする場合は、次の点を考慮してください。

- Wavelength Zones 内の Amazon EC2 インスタンスは、アベイラビリティーゾーン内の Amazon EC2 インスタンスよりも高コストです。
- AWS Wavelength Zones で実行されているアプリケーションとエンドユーザーの間の遅延は低くなります。一部のワークロードでは、遅延の影響が発生します。たとえば、Wavelength Zones とアベイラビリティーゾーンの間で Ingress トラフィックが混在している場合などです。

重要

通常、Wavelength Zones 内の Amazon EC2 インスタンスとリージョン内の Amazon EC2 インスタンス間の最大伝送単位 (MTU) は 1300 です。クラスターネットワークの MTU は、オーバーヘッドを考慮して、常に EC2 の MTU よりも小さくする必要があります。具体的なオーバーヘッドは、ネットワークプラグインによって決まります。たとえば、OVN-Kubernetes のオーバーヘッドは **100 bytes** です。

ネットワークプラグインは、IPsec などの追加機能を提供できます。MTU のサイズには、このような追加機能も影響します。

詳細は、AWS ドキュメントの How AWS Wavelength work を参照してください。

OpenShift Container Platform 4.12 で、リモートゾーンで使用するために設計された新しいコンピュートプールの **エッジ** が導入されました。エッジコンピュートプールの設定は、Wavelength Zones の場所間で共通です。Wavelength Zones リソース上の EC2 や EBS などのリソースのタイプとサイズの制限により、デフォルトのインスタンスタイプが従来のコンピュートプールと異なる場合があります。

Wavelength Zones の場所のデフォルト Elastic Block Store (EBS) は **gp2** であり、非エッジコンピュートプールとは異なります。各 Wavelength Zones に使用される、エッジコンピュートプールのインスタンスタイプも、ゾーンのインスタンスオファリングに応じて、他のコンピュートプールと異なる場合があります。

エッジコンピュートプールは、開発者が AWS Wavelength Zones ノードにアプリケーションをデプロイするために使用できる新しいラベルを作成します。新しいラベルは次のとおりです。

- **node-role.kubernetes.io/edge=''**
- **machine.openshift.io/zone-type=wavelength-zone**
- **machine.openshift.io/zone-group=$ZONE_GROUP_NAME**

デフォルトでは、エッジコンピュートプールのマシンセットは **NoSchedule** テイントを定義して、Wavelength Zones インスタンスに他のワークロードが拡散するのを防ぎます。ユーザーは、Pod 仕様で容認を定義している場合にのみユーザーワークロードを実行できます。

関連情報

- MTU 値の選択
- クラスターネットワークの MTU 変更
- テイントおよび容認 (Toleration) について
- ストレージクラス
- Ingress コントローラーのシャード化

6.3.12.3. インストールの要件

AWS Wavelength Zones 環境にクラスターをインストールする前に、Wavelength Zone 機能を導入できるようにインフラストラクチャーを設定する必要があります。

6.3.12.3.1. AWS Wavelength Zones へのオプトイン

AWS Wavelength Zones でサブネットを作成する予定がある場合は、各ゾーングループに個別にオプトインする必要があります。

前提条件

- AWS CLI をインストールしている。
- OpenShift Container Platform クラスターをデプロイする AWS リージョンを決定しました。
- ゾーングループにオプトインするユーザーまたはロールアカウントに、寛容な IAM ポリシーをアタッチしました。

手順

1. 次のコマンドを実行して、AWS リージョンで利用可能なゾーンをリスト表示します。

   AWS リージョンで利用可能な AWS Wavelength Zones をリストするコマンドの例

   ```
   $ aws --region "<value_of_AWS_Region>" ec2 describe-availability-zones \
   ```

```
    --query 'AvailabilityZones[].[{ZoneName: ZoneName, GroupName: GroupName, Status:
OptInStatus}]' \
    --filters Name=zone-type,Values=wavelength-zone \
    --all-availability-zones
```

AWS リージョンによっては、利用可能なゾーンのリストが長くなる場合があります。このコマンドは次のフィールドを返します。

ZoneName

Wavelength Zones の名前。

GroupName

ゾーンを設定するグループ。リージョンにオプトインするには、この名前を保存しておきます。

Status

Wavelength Zones グループのステータス。ステータスが **not-opted-in** の場合は、次の手順で説明するように **GroupName** をオプトインする必要があります。

2. 次のコマンドを実行して、AWS アカウントのゾーングループにオプトインします。

```
$ aws ec2 modify-availability-zone-group \
    --group-name "<value_of_GroupName>" \ 1
    --opt-in-status opted-in
```

1 **<value_of_GroupName>** は、サブネットを作成する Wavelength Zones のグループの名前に置き換えます。たとえば、Wavelength Zones としてゾーン **us-east-1-wl1-nyc-wlz-1** (米国東部ニューヨーク) を使用するには、**us-east-1-wl1** を指定します。

6.3.12.3.2. AWS Marketplace イメージの取得

AWS Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に AWS を通じてサブスクライブする必要があります。オファーにサブスクライブすると、インストールプログラムがコンピュートノードのデプロイに使用する AMI ID が提供されます。

前提条件

- オファーを購入するための AWS アカウントを持っている。このアカウントは、クラスターのインストールに使用されるアカウントと同じである必要はありません。

手順

1. AWS Marketplace で OpenShift Container Platform サブスクリプションを完了します。
2. ご使用の AWS リージョンの AMI ID を記録します。インストールプロセスの一環として、クラスターをデプロイする前に、この値で **install-config.yaml** ファイルを更新する必要があります。

AWS Marketplace コンピュートノードを含む install-config.yaml ファイルのサンプル

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
```

```
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 1
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 2
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

1 AWS Marketplace サブスクリプションの AMI ID。

2 AMI ID は特定の AWS リージョンに関連付けられています。インストール設定ファイルを作成するときは、サブスクリプションの設定時に指定したものと同じ AWS リージョンを選択してください。

6.3.12.4. インストールの準備

ノードを Wavelength Zone に拡張する前に、クラスターのインストール環境用に特定のリソースを準備する必要があります。

6.3.12.4.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表6.20 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと

IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

6.3.12.4.2. AWS のテスト済みインスタンスタイプ

次の Amazon Web Services (AWS) インスタンスタイプは、AWS Wavelength Zones で使用できることを OpenShift Container Platform でテスト済みです。

注記

AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

例6.51 AWS Wavelength Zones で使用できる 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **r5.***
- **t3.***

関連情報

- AWS ドキュメントの AWS Wavelength features を参照してください。

6.3.12.4.3. インストール設定ファイルの作成

インストールプログラムがクラスターをデプロイするために必要なインストール設定ファイルを生成し、カスタマイズします。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- Red Hat が公開している付属の Red Hat Enterprise Linux CoreOS (RHCOS) AMI のある AWS リージョンにクラスターをデプロイしようとしている。カスタム AMI が必要な AWS リージョン (AWS GovCloud リージョンなど) にデプロイする場合は、**install-config.yaml** ファイルを手動で作成する必要があります。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **aws** を選択します。
      iii. AWS プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

注記

AWS アクセスキー ID およびシークレットアクセスキーは、インストールホストの現行ユーザーのホームディレクトリーの **~/.aws/credentials** に保存されます。エクスポートされたプロファイルの認証情報がファイルにない場合は、インストールプログラムにより認証情報の入力が求めるプロンプトが出されます。インストールプログラムに指定する認証情報は、ファイルに保存されます。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. オプション: **install-config.yaml** ファイルをバックアップします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

6.3.12.4.4. エッジコンピュートプールを含むインストール設定ファイルの例

次の例は、エッジマシンプール設定を含む **install-config.yaml** ファイルを示しています。

カスタムインスタンスタイプのエッジプールを使用する設定

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
compute:
- name: edge
  platform:
    aws:
      type: r5.2xlarge
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

インスタンスのタイプは場所によって異なります。クラスターを実行する Wavelength Zones で利用可能かどうかを確認するには、AWS のドキュメントを参照してください。

カスタムセキュリティーグループを持つエッジプールを使用する設定

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
```

```
compute:
- name: edge
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 1
        - sg-2
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

1 Amazon EC2 コンソールに表示されるセキュリティーグループの名前を指定します。必ず **sg** 接頭辞を含めてください。

6.3.12.5. AWS Wavelength Zones 環境のクラスターインストール方法

以下のいずれかのインストール方法を選択して、Wavelength Zones に定義されたエッジコンピュートノードを使用して OpenShift Container Platform クラスターを AWS にインストールします。

- 完全に自動化された方法: クラスターをインストールして、コンピュートノードをエッジコンピュートプールに迅速に拡張します。インストールプログラムが、OpenShift Container Platform クラスター用のインフラストラクチャーリソースを自動的に作成します。
- 既存の VPC を使用する方法: AWS 上のクラスターを既存の VPC にインストールします。この場合、Wavelength Zones サブネットを **install-config.yaml** ファイルに指定します。

次のステップ

次のいずれかの方法を選択して、OpenShift Container Platform クラスターを AWS Wavelength Zones 環境にインストールします。

- クラスターの AWS Wavelength Zones へのクイックインストール
- AWS Wavelength Zones を使用するためのインストール設定ファイルの変更

6.3.12.6. クラスターの AWS Wavelength Zones へのクイックインストール

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS) にクラスターをすばやくインストールして、コンピュートノードを Wavelength Zones の場所に拡張できます。このインストール方法を使用すると、設定ファイルで定義した各ゾーンのネットワークリソースと Wavelength Zones サブネットが、インストールプログラムによって自動的に作成されます。インストールをカスタマイズするには、クラスターをデプロイする前に、**install-config.yaml** ファイル内のパラメーターを変更する必要があります。

6.3.12.6.1. AWS Wavelength Zones を使用するためのインストール設定ファイルの変更

install-config.yaml ファイルを変更して、AWS Wavelength Zones を含めます。

前提条件

- AWS アカウントが設定されている。

- **aws configure** を実行して、AWS キーと AWS リージョンをローカル AWS プロファイルに追加している。
- OpenShift Container Platform クラスターのサブネットを自動的に作成するようにインストールプログラムを指定する際に適用される設定上の制限を理解している。
- 各ゾーンの Wavelength Zones グループにオプトインしている。
- 「インストール設定ファイルの作成」手順を使用して、**install-config.yaml** ファイルを作成している。

手順

1. **install-config.yaml** ファイルを変更して、エッジコンピュートプールの **platform.aws.zones** プロパティーで Wavelength Zones 名を指定します。

```
# ...
platform:
  aws:
    region: <region_name> 1
compute:
- name: edge
  platform:
    aws:
      zones: 2
      - <wavelength_zone_name>
#...
```

1 AWS リージョン名。

2 使用する Wavelength Zones 名のリストは、**platform.aws.region** フィールドで指定した同じ AWS リージョンに存在する必要があります。

エッジノードを Los Angeles と Las Vegas の Wavelength Zones に拡張する us-west-2 AWS リージョンにクラスターをインストールする設定の例

```
apiVersion: v1
baseDomain: example.com
metadata:
  name: cluster-name
platform:
  aws:
    region: us-west-2
compute:
- name: edge
  platform:
    aws:
      zones:
      - us-west-2-wl1-lax-wlz-1
      - us-west-2-wl1-las-wlz-1
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
#...
```

2. クラスターをデプロイします。

関連情報

- インストール設定ファイルの作成
- AWS Wavelength Zones のクラスターの制限

次のステップ

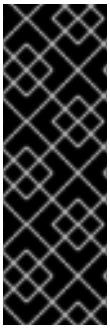
- クラスターのデプロイ

6.3.12.7. Wavelength Zone のサブネットを持つ既存の VPC へのクラスターのインストール

クラスターを Amazon Web Services (AWS) 上の既存の Amazon Virtual Private Cloud (VPC) にインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

AWS 上のクラスターを既存の VPC にインストールするには、AWS Wavelength Zones を使用してコンピュートノードをクラウドインフラストラクチャーのエッジまで拡張する必要があります。

このドキュメントの CloudFormation テンプレートを使用して、ネットワークリソースを作成できます。さらに、テンプレートを変更してインフラストラクチャーをカスタマイズしたり、テンプレートに含まれる情報を使用して会社のポリシーに基づいて AWS リソースを作成したりできます。



重要

インストーラによってプロビジョニングされたインフラストラクチャーのインストールを実行する手順は、例としてのみ提供されています。既存の VPC にクラスターをインストールするには、クラウドプロバイダーと OpenShift Container Platform のインストールプロセスに関する知識が必要です。CloudFormation テンプレートを使用すると、これらの手順の完了を支援したり、独自のクラスターのインストールをモデル化したりできます。CloudFormation テンプレートを使用してリソースを作成する代わりに、これらのリソースを生成するために他の方法を使用することを決定できます。

6.3.12.7.1. AWS での VPC の作成

OpenShift Container Platform クラスター用の Amazon Web Services (AWS) で、Virtual Private Cloud (VPC) を作成し、Wavelength Zones のすべての場所にサブネットを作成すると、コンピュートノードをエッジロケーションに拡張できます。VPC は、VPN やルートテーブルなどの要件に合わせてさらにカスタマイズできます。初期デプロイメントに含まれていない新しい Wavelength Zones サブネットを追加することもできます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、VPC を表す AWS リソースのスタックを作成できます。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーと AWS リージョンをローカル AWS プロファイルに追加している。
- AWS アカウントで AWS Wavelength Zones にオプトインしている。

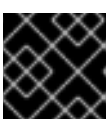
手順

1. CloudFormation テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "3" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1. VPC の CIDR ブロック。
2. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
3. VPC をデプロイするアベイラビリティーゾーンの数。
4. **1** から **3** の間の整数を指定します。
5. 各アベイラビリティーゾーン内の各サブネットのサイズ。
6. **5** から **13** の間の整数を指定します。 ここで、**5** は **/27** であり、 **13** は **/19** です。

2. このドキュメントの「VPC の CloudFormation テンプレート」セクションに移動し、記載されているテンプレートから構文をコピーします。コピーしたテンプレートの構文を YAML ファイルとしてローカルシステムに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
3. 次のコマンドを実行して、CloudFormation テンプレートを起動し、VPC を表す AWS リソースのスタックを作成します。



重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> \ 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1. **<name>** は **cluster-vpc** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーターの JSON ファイルの相対パスと名前です。

出力例

```
arn:aws:cloudformation:us-east-1:123456789012:stack/cluster-vpc/dbedae40-2fd3-11eb-820e-12a48460849f
```

4. 次のコマンドを実行して、テンプレートコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーター値を、クラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

VpcId	VPC の ID。
PublicSubnetIds	新規パブリックサブネットの ID。
PrivateSubnetIds	新規プライベートサブネットの ID。
PublicRouteTableId	新しいパブリックルートテーブル ID の ID。

6.3.12.7.2. VPC の CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例6.52 VPC の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
```

```
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 =
/19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
```

```
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
```

```
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
```

```
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP2
      - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
    AllocationId:
      "Fn::GetAtt":
```

```
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
```

```
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
  PrivateRouteTableIds:
    Description: Private Route table IDs
    Value:
      !Join [
        ",",
        [
          !Join ["=", [
            !Select [0, "Fn::GetAZs": !Ref "AWS::Region"],
            !Ref PrivateRouteTable
          ]],
          !If [DoAz2,
              !Join ["=", [!Select [1, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable2]],
              !Ref "AWS::NoValue"
          ],
          !If [DoAz3,
              !Join ["=", [!Select [2, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable3]],
              !Ref "AWS::NoValue"
          ]
        ]
      ]
```

6.3.12.7.3. VPC キャリアーゲートウェイの作成

Wavelength Zones で実行される OpenShift Container Platform クラスターでパブリックサブネットを使用するには、キャリアーゲートウェイを作成し、キャリアーゲートウェイを VPC に関連付ける必要があります。サブネットは、ロードバランサーまたはエッジコンピュートノードをデプロイするのに役立ちます。

OpenShift Container Platform クラスター用の Wavelength Zones の場所に、エッジノードやインターネットに接続されたロードバランサーを作成するには、以下の必要なネットワークコンポーネントを作成する必要があります。

- 既存の VPC に関連付けるキャリアーゲートウェイ
- ルートエントリーをリストするキャリールートテーブル
- キャリアールートテーブルに関連付けるサブネット

キャリアーゲートウェイは、Wavelength Zone 内のサブネットのみを含む VPC に存在します。

以下では、AWS Wavelength Zones の場所に関連するキャリアーゲートウェイの機能を説明します。

- Wavelength Zone とキャリアーネットワーク (キャリアーネットワークから利用可能なデバイスを含む) の間の接続を提供します。
- ネットワークボーダーグループに格納されているパブリック IP アドレスである IP アドレスを Wavelength Zones からキャリアー IP アドレスに変換するなど、ネットワークアドレス変換

(NAT) 機能を実行します。このような変換機能は、受信トラフィックと送信トラフィックに適用されます。

- 特定の場所にあるキャリアーネットワークからの受信トラフィックを許可します。
- キャリアーネットワークとインターネットへの送信トラフィックを許可します。

注記

インターネットからキャリアーゲートウェイを経由した Wavelength Zone への受信接続設定は存在しません。

このドキュメントの CloudFormation テンプレートを使用して、次の AWS リソースのスタックを作成できます。

- テンプレート内の VPC ID に関連付ける 1 つのキャリアーゲートウェイ
- **<ClusterName>-public-carrier** という名前の Wavelength Zone の 1 つのパブリックルートテーブル
- キャリアーゲートウェイをターゲットとする新しいルートテーブルのデフォルトの IPv4 ルートエントリー
- AWS Simple Storage Service (S3) の VPC ゲートウェイエンドポイント

注記

このドキュメントの CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、記載されている情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。

手順

1. ドキュメントの次のセクション「VPC キャリアーゲートウェイ用の CloudFormation テンプレート」に移動し、**VPC キャリアーゲートウェイ用の CloudFormation テンプレート** から構文をコピーします。コピーしたテンプレートの構文を YAML ファイルとしてローカルシステムに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. 次のコマンドを実行して CloudFormation テンプレートをデプロイします。これにより、VPC を表す AWS リソースのスタックが作成されます。

```
$ aws cloudformation create-stack --stack-name <stack_name> \ 1
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ 2
```

```
--parameters \//
 ParameterKey=VpcId,ParameterValue="${VpcId}" \ 3
 ParameterKey=ClusterName,ParameterValue="${ClusterName}" 4
```

1. **<stack_name>** は CloudFormation スタックの名前です (例: **clusterName-vpc-carrier-gw**)。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルの相対パスと名前です。
3. **<VpcId>** は、「AWS での VPC の作成」セクションで作成した CloudFormation スタックの出力から抽出した VPC ID です。
4. **<ClusterName>** は、CloudFormation スタックによって作成されるリソースに接頭辞として付加するカスタム値です。**install-config.yaml** 設定ファイルの **metadata.name** セクションで定義されているのと同じ名前を使用できます。

出力例

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-2fd3-11eb-820e-12a48460849f
```

検証

- 次のコマンドを実行して、CloudFormation テンプレートコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <stack_name>
```

StackStatus に **CREATE_COMPLETE** が表示されると、出力に次のパラメーターの値が表示されます。このパラメーター値を、クラスターを作成するために実行する他の CloudFormation テンプレートに必ず指定してください。

PublicRouteTableId	キャリアーインフラストラクチャーのルートテーブルの ID。

関連情報

- AWS ドキュメントの Amazon S3 を参照してください。

6.3.12.7.4. VPC キャリアーゲートウェイ用の CloudFormation テンプレート

次の CloudFormation テンプレートを使用して、AWS Wavelength インフラストラクチャーにキャリアーゲートウェイをデプロイできます。

例6.53 VPC キャリアーゲートウェイ用の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Creating Wavelength Zone Gateway (Carrier Gateway).

Parameters:
  VpcId:
```

```
    Description: VPC ID to associate the Carrier Gateway.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster Name or Prefix name to prepend the tag Name for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.

Resources:
  CarrierGateway:
    Type: "AWS::EC2::CarrierGateway"
    Properties:
      VpcId: !Ref VpcId
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "cagw"]]

  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VpcId
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "public-carrier"]]

  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: CarrierGateway
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      CarrierGatewayId: !Ref CarrierGateway

  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VpcId
```

```
Outputs:
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
```

6.3.12.7.5. Wavelength Zones のサブネットの作成

OpenShift Container Platform クラスターのエッジコンピュートノードのマシンセットを設定する前に、Wavelength Zones にサブネットを作成する必要があります。コンピュートノードをデプロイする Wavelength Zone ごとに次の手順を実行してください。

このドキュメントの CloudFormation テンプレートを使用して、CloudFormation スタックを作成できます。その後、このスタックを使用してサブネットをカスタムプロビジョニングできます。

注記

このドキュメントの CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、記載されている情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- Wavelength Zones グループにオプトインしている。

手順

1. このドキュメントの「VPC サブネット用の CloudFormation テンプレート」セクションに移動し、テンプレートから構文をコピーします。コピーしたテンプレートの構文を YAML ファイルとしてローカルシステムに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. 次のコマンドを実行して CloudFormation テンプレートをデプロイします。これにより、VPC を表す AWS リソースのスタックが作成されます。

```
$ aws cloudformation create-stack --stack-name <stack_name> \ 1
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ 2
  --parameters \
    ParameterKey=VpcId,ParameterValue="${VPC_ID}" \ 3
    ParameterKey=ClusterName,ParameterValue="${CLUSTER_NAME}" \ 4
    ParameterKey=ZoneName,ParameterValue="${ZONE_NAME}" \ 5
    ParameterKey=PublicRouteTableId,ParameterValue="${ROUTE_TABLE_PUB}" \ 6
    ParameterKey=PublicSubnetCidr,ParameterValue="${SUBNET_CIDR_PUB}" \ 7
    ParameterKey=PrivateRouteTableId,ParameterValue="${ROUTE_TABLE_PVT}" \ 8
    ParameterKey=PrivateSubnetCidr,ParameterValue="${SUBNET_CIDR_PVT}" 9
```

1. **<stack_name>** は、CloudFormation スタックの名前です (**cluster-wl-<wavelength_zone_shortname>** など)。クラスターを削除する場合に、このスタックの
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルの相対パスと名前です。
3. **${VPC_ID}** は VPC ID であり、VPC 用の CloudFormation テンプレートの出力に含まれる値 **VpcID** です。
4. **${ZONE_NAME}** は、サブネットを作成する Wavelength Zones 名の値です。
5. **${CLUSTER_NAME}** は、新しい AWS リソース名の接頭辞として使用する **ClusterName** の値です。
6. **${ROUTE_TABLE_PUB}** は、VPC のキャリアーゲートウェイ CloudFormation スタックの出力から抽出した **PublicRouteTableId** です。
7. **${SUBNET_CIDR_PUB}** は、パブリックサブネットの作成に使用する有効な CIDR ブロックです。このブロックは、VPC CIDR ブロック **VpcCidr** の一部である必要があります。
8. **${ROUTE_TABLE_PVT}** は、VPC の CloudFormation スタックの出力から抽出した **PrivateRouteTableId** です。
9. **${SUBNET_CIDR_PVT}** は、プライベートサブネットの作成に使用する有効な CIDR ブロックです。このブロックは、VPC CIDR ブロック **VpcCidr** の一部である必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-820e-11eb-2fd3-12a48460849f
```

検証

- 次のコマンドを実行して、テンプレートコンポーネントが存在することを確認します。

  ```
  $ aws cloudformation describe-stacks --stack-name <stack_name>
  ```

  StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーター値を、クラスターを作成するために実行する他の CloudFormation テンプレートに必ず指定してください。

PublicSubnetId	CloudFormation スタックによって作成されたパブリックサブネットの ID。
PrivateSubnetId	CloudFormation スタックによって作成されたプライベートサブネットの ID。

6.3.12.7.6. VPC サブネット用の CloudFormation テンプレート

次の CloudFormation テンプレートを使用して、Wavelength Zones インフラストラクチャー上のゾーンにプライベートサブネットとパブリックサブネットをデプロイできます。

■

例6.54 VPC サブネット用の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice Subnets (Public and Private)

Parameters:
  VpcId:
    Description: VPC ID that comprises all the target subnets.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster name or prefix name to prepend the Name tag for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.
  ZoneName:
    Description: Zone Name to create the subnets, such as us-west-2-lax-1a.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ZoneName parameter must be specified.
  PublicRouteTableId:
    Description: Public Route Table ID to associate the public subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PublicRouteTableId parameter must be specified.
  PublicSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for public subnet.
    Type: String
  PrivateRouteTableId:
    Description: Private Route Table ID to associate the private subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PrivateRouteTableId parameter must be specified.
  PrivateSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for private subnet.
    Type: String


Resources:
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PublicSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
```

```
      Value: !Join ['-', [!Ref ClusterName, "public", !Ref ZoneName]]

  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTableId

  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PrivateSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "private", !Ref ZoneName]]

  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTableId

Outputs:
  PublicSubnetId:
    Description: Subnet ID of the public subnets.
    Value:
      !Join ["", [!Ref PublicSubnet]]

  PrivateSubnetId:
    Description: Subnet ID of the private subnets.
    Value:
      !Join ["", [!Ref PrivateSubnet]]
```

6.3.12.7.7. AWS Wavelength Zones サブネットを使用するためのインストール設定ファイルの変更

install-config.yaml ファイルを変更して、Wavelength Zones のサブネットを含めます。

前提条件

- 「Wavelength Zone のサブネットの作成」手順を使用して、サブネットを作成している。
- 「インストール設定ファイルの作成」手順を使用して、**install-config.yaml** ファイルを作成している。

手順

- **install-config.yaml** 設定ファイルを変更して、**platform.aws.subnets** パラメーターで Wavelength Zones のサブネットを指定します。

  Wavelength Zones のサブネットを含むインストール設定ファイルの例

  ```
  # ...
  ```

```
platform:
  aws:
    region: us-west-2
    subnets: (1)
    - publicSubnetId-1
    - publicSubnetId-2
    - publicSubnetId-3
    - privateSubnetId-1
    - privateSubnetId-2
    - privateSubnetId-3
    - publicOrPrivateSubnetID-Wavelength-1
# ...
```

(1) ゾーン (アベイラビリティーゾーンと Wavelength Zones) 内に作成したサブネット ID のリスト。

関連情報

- 作成した CloudFormation スタックを表示する方法の詳細は、AWS CloudFormation コンソール を参照してください。
- AWS プロファイルと認証情報を設定する方法の詳細は、AWS ドキュメントの Configuration and credential file settings を参照してください。

次のステップ

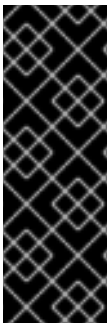
- クラスターのデプロイ

6.3.12.8. オプション: パブリック IP アドレスをエッジコンピュートノードに割り当てます。

ワークロードによっては、Wavelength Zones インフラストラクチャー上のパブリックサブネットにエッジコンピュートノードをデプロイすることが必要になります。その場合は、クラスターのインストール時にマシンセットマニフェストを設定できます。

AWS Wavelength Zones インフラストラクチャーは、指定されたゾーン内のネットワークトラフィックにアクセスします。そのため、そのゾーンに近いエンドユーザーにサービスを提供する際に、アプリケーションで低遅延を利用できます。

デフォルト設定は、プライベートサブネットにコンピュートノードをデプロイする設定であり、お客様のニーズに合わない可能性があります。そのため、インフラストラクチャーにさらにカスタマイズを適用する必要がある場合は、パブリックサブネットにエッジコンピュートノードを作成することを検討してください。



重要

デフォルトでは、OpenShift Container Platform はプライベートサブネットにコンピュートノードをデプロイします。最高のパフォーマンスを得るには、パブリック IP アドレスがサブネットに接続されているサブネットにコンピュートノードを配置することを検討してください。

追加のセキュリティーグループを作成する必要があります。ただし、インターネット経由でグループのルールを開くのは、本当に必要な場合だけに留めてください。

手順

1. インストールプログラムが含まれるディレクトリーに移動し、マニフェストファイルを生成します。インストールマニフェストが **openshift** および **manifests** ディレクトリーレベルに作成されていることを確認します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. インストールプログラムが Wavelength Zones 用に生成するマシンセットマニフェストを編集して、マニフェストがパブリックサブネットにデプロイされるようにします。**spec.template.spec.providerSpec.value.publicIP** パラメーターに **true** を指定します。

クラスターを Wavelength Zones に迅速にインストールするためのマシンセットマニフェスト設定の例

```
spec:
  template:
    spec:
      providerSpec:
        value:
          publicIp: true
          subnet:
            filters:
              - name: tag:Name
                values:
                  - ${INFRA_ID}-public-${ZONE_NAME}
```

Wavelength Zones サブネットを持つ既存の VPC にクラスターをインストールするためのマシンセットマニフェスト設定の例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <infrastructure_id>-edge-<zone>
  namespace: openshift-machine-api
spec:
  template:
    spec:
      providerSpec:
        value:
          publicIp: true
```

6.3.12.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、無効にします。

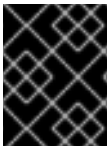
注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.3.12.10. デプロイしたクラスターのステータスの確認

OpenShift Container Platform が AWS Wavelength Zones に正常にデプロイされたことを確認します。

6.3.12.10.1. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.3.12.10.2. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

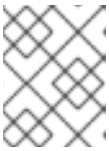
```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- Web コンソールへのアクセス

6.3.12.10.3. エッジコンピューティングプールで作成されたノードの検証

AWS Wavelength Zones インフラストラクチャーを使用するクラスターをインストールしたら、インストール時に作成したマシンセットマニフェストによって作成されたマシンのステータスを確認します。

1. **install-config.yaml** ファイルに追加したサブネットから作成されたマシンセットを確認するには、次のコマンドを実行します。

```
$ oc get machineset -n openshift-machine-api
```

出力例

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
cluster-7xw5g-edge-us-east-1-wl1-nyc-wlz-1   1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1a              1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1b              1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1c              1         1         1       1           3h4m
```

2. マシンセットから作成されたマシンを確認するには、次のコマンドを実行します。

```
$ oc get machines -n openshift-machine-api
```

出力例

```
NAME                                      PHASE     TYPE        REGION      ZONE            AGE
cluster-7xw5g-edge-us-east-1-wl1-nyc-wlz-1-wbclh   Running   c5d.2xlarge   us-east-1   us-east-1-wl1-nyc-wlz-1   3h
cluster-7xw5g-master-0                    Running   m6i.xlarge    us-east-1   us-east-1a      3h4m
cluster-7xw5g-master-1                    Running   m6i.xlarge    us-east-1   us-east-1b      3h4m
cluster-7xw5g-master-2                    Running   m6i.xlarge    us-east-1   us-east-1c      3h4m
cluster-7xw5g-worker-us-east-1a-rtp45     Running   m6i.xlarge    us-east-1   us-east-1a      3h
cluster-7xw5g-worker-us-east-1b-glm7c     Running   m6i.xlarge    us-east-1   us-east-1b      3h
cluster-7xw5g-worker-us-east-1c-qfvz4     Running   m6i.xlarge    us-east-1   us-east-1c      3h
```

3. エッジロールを持つノードを確認するには、次のコマンドを実行します。

```
$ oc get nodes -l node-role.kubernetes.io/edge
```

出力例

```
NAME                          STATUS   ROLES        AGE    VERSION
ip-10-0-207-188.ec2.internal   Ready    edge,worker   172m   v1.25.2+d2e245f
```

次のステップ

- インストールを検証 します。
- 必要に応じて、リモートヘルスをオプトアウト できます。

6.3.13. AWS Outposts 上のコンピュートノードを使用してクラスターをインストールする

OpenShift Container Platform バージョン 4.14 では、テクノロジープレビュー機能として、AWS Outposts で実行されているコンピュートノードを使用して、Amazon Web Services (AWS) にクラスターをインストールすることができました。OpenShift Container Platform バージョン 4.15 以降、このインストール方法はサポートされなくなりました。

代わりに、AWS 上のクラスターを既存の VPC にインストール し、インストール後の設定タスクとして AWS Outposts にコンピュートノードをプロビジョニングできます。

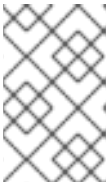
詳細は、AWS VPC クラスターの AWS Outpost への拡張 を参照してください。

6.4. USER-PROVISIONED INFRASTRUCTURE

6.4.1. AWS にクラスターをインストールする準備

以下の手順を実行して、AWS に OpenShift Container Platform クラスターをインストールする準備をします。

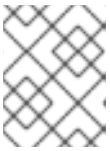
- クラスターのインターネット接続を検証します。
- AWS アカウントを設定します。
- インストールプログラムをダウンロードします。



注記

非接続環境にインストールする場合は、ミラーリングしたコンテンツからインストールプログラムを抽出します。詳細は、非接続インストール用のイメージのミラーリング を参照してください。

- OpenShift CLI (**oc**) をインストールします。



注記

非接続環境にインストールする場合は、ミラーホストに **oc** をインストールします。

- SSH キーペアを生成します。OpenShift Container Platform クラスターのデプロイ後にこのキーペアを使用して、クラスターのノードに対する認証を行うことができます。
- user-provisioned infrastructure を準備します。
- クラウドアイデンティティーおよびアクセス管理 (IAM) API が環境内でアクセスできない場合、または管理者レベルの認証情報シークレットを **kube-system** namespace に保存しない場合は、AWS の長期認証情報を手動で作成 するか、Amazon Web Services Security Token Service (AWS STS) で 短期認証情報を使用するように AWS クラスターを設定 します。

6.4.1.1. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。

- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.4.1.2. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

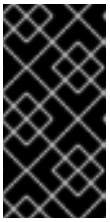
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードしま

す。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

6.4.1.3. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。

2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

6.4.1.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラ

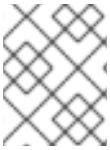
ムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの ~/**.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

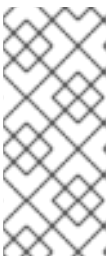
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、**~/.ssh/id_rsa** および **~/.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.4.1.5. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

6.4.2. user-provisioned infrastructure の vSphere インストール要件

プロビジョニングしたインフラストラクチャーへのインストールを開始する前に、vSphere 環境が次のインストール要件を満たしていることを確認してください。

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

6.4.2.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表6.21 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

6.4.2.1.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

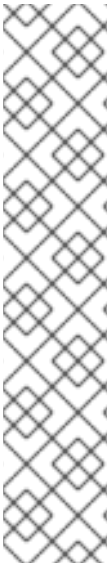
表6.22 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

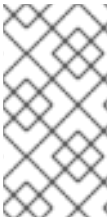
プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

6.4.2.1.2. AWS のテスト済みインスタンスタイプ

以下の Amazon Web Services(AWS)インスタンスタイプは OpenShift Container Platform でテストされています。



注記

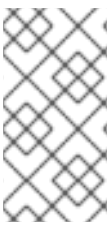
AWS インスタンスには、次の表に記載されているマシンタイプを使用してください。表に記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、「クラスターインストールの最小リソース要件」セクションに記載されている最小リソース要件と一致していることを確認してください。

例6.55 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

6.4.2.1.3. 64 ビット ARM インフラストラクチャー上の AWS のテスト済みインスタンスタイプ

次の Amazon Web Services (AWS) 64 ビット ARM インスタンスタイプは、OpenShift Container Platform でテストされています。



注記

AWS ARM インスタンスには、次のチャートに含まれるマシンタイプを使用してください。チャートに記載されていないインスタンスタイプを使用する場合は、使用するインスタンスサイズが、クラスターインストールの最小リソース要件に記載されている最小リソース要件と一致していることを確認してください。

例6.56 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

6.4.2.2. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

6.4.2.3. 必要な AWS インフラストラクチャーコンポーネント

OpenShift Container Platform を Amazon Web Services (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、マシンとサポートするインフラストラクチャーの両方を手動で作成する必要があります。

各種プラットフォームの統合テストの詳細については、OpenShift Container Platform 4.x のテスト済みインテグレーション のページを参照してください。

提供される CloudFormation テンプレートを使用すると、以下のコンポーネントを表す AWS リソースのスタックを作成できます。

- AWS Virtual Private Cloud (VPC)
- ネットワークおよび負荷分散コンポーネント
- セキュリティーグループおよびロール
- OpenShift Container Platform ブートストラップノード
- OpenShift Container Platform コントロールプレーンノード
- OpenShift Container Platform コンピュートノード

または、コンポーネントを手動で作成するか、クラスターの要件を満たす既存のインフラストラクチャーを再利用できます。コンポーネントの相互関係についての詳細は、CloudFormation テンプレートを参照してください。

6.4.2.3.1. 他のインフラストラクチャーコンポーネント

- 1 つの VPC
- DNS エントリー
- ロードバランサー (classic または network) およびリスナー
- パブリックおよびプライベート Route 53 ゾーン
- セキュリティーグループ

- IAM ロール
- S3 バケット

非接続環境で作業している場合、EC2、ELB、および S3 エンドポイントのパブリック IP アドレスに到達することはできません。インストール中にインターネットトラフィックを制限するレベルに応じて、次の設定オプションを使用できます。

オプション 1: VPC エンドポイントを作成する
VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

このオプションを使用すると、VPC および必要な AWS サービスの間でネットワークトラフィックがプライベートのままになります。

オプション 2: VPC エンドポイントなしでプロキシーを作成する
インストールプロセスの一環として、HTTP または HTTPS プロキシーを設定できます。このオプションを使用すると、インターネットトラフィックはプロキシーを経由して、必要な AWS サービスに到達します。

オプション 3: VPC エンドポイントでプロキシーを作成する
インストールプロセスの一環として、VPC エンドポイントを使用して HTTP または HTTPS プロキシーを設定できます。VPC エンドポイントを作成し、クラスターが使用しているサブネットにアタッチします。次のようにエンドポイントに名前を付けます。

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

install-config.yaml ファイルでプロキシーを設定するときに、これらのエンドポイントを **noProxy** フィールドに追加します。このオプションを使用すると、プロキシーはクラスターがインターネットに直接アクセスするのを防ぎます。ただし、VPC と必要な AWS サービスの間のネットワークトラフィックはプライベートのままです。

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。

必要な DNS および負荷分散コンポーネント

DNS およびロードバランサー設定では、パブリックホストゾーンを使用する必要があり、クラスターのインフラストラクチャーをプロビジョニングする場合にインストールプログラムが使用するものと同様のプライベートホストゾーンを使用できます。ロードバランサーに解決する DNS エントリーを作成する必要があります。**api.<cluster_name>.<domain>** のエントリーは外部ロードバランサーを参照し、**api-int.<cluster_name>.<domain>** のエントリーは内部ロードバランサーを参照する必要があります。

またクラスターには、Kubernetes API とその拡張に必要なポート 6443、および新規マシンの Ignition 設定ファイルに必要なポート 22623 のロードバランサーおよびリスナーが必要です。ターゲットはコントロールプレーンノードになります。ポート 6443 はクラスター外のクライアントとクラスター内のノードからもアクセスできる必要があります。ポート 22623 はクラスター内のノードからアクセスできる必要があります。

コンポーネント	AWS タイプ	説明
DNS	**AWS::Route53::HostedZone**	内部 DNS のホストゾーン。
パブリックロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	パブリックサブネットのロードバランサー。
外部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	外部 API サーバーのエイリアスレコード。
外部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	外部ロードバランサー用のポート 6443 のリスナー。
外部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	外部ロードバランサーのターゲットグループ。

コンポーネント	AWS タイプ	説明
プライベートロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	プライベートサブネットのロードバランサー。
内部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	内部 API サーバーのエイリアスレコード。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサー用のポート 22623 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサーのポート 6443 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。

セキュリティーグループ

コントロールプレーンおよびワーカーマシンには、以下のポートへのアクセスが必要です。

グループ	タイプ	IP プロトコル	ポート範囲
MasterSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
		tcp	**6443**
		tcp	**22623**
WorkerSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**

グループ	タイプ	IP プロトコル	ポート範囲
BootstrapSecurityGroup	**AWS::EC2::SecurityGroup**	**tcp**	**22**
		tcp	**19531**

コントロールプレーンの Ingress

コントロールプレーンマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressEtcd	etcd	**tcp**	**2379- 2380**
MasterIngressVxlan	Vxlan パケット	**udp**	**4789**
MasterIngressWorkerVxlan	Vxlan パケット	**udp**	**4789**
MasterIngressInternal	内部クラスター通信および Kubernetes プロキシーメトリック	**tcp**	**9000 - 9999**
MasterIngressWorkerInternal	内部クラスター通信	**tcp**	**9000 - 9999**
MasterIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250 - 10259**
MasterIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250 - 10259**
MasterIngressIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
MasterIngressWorkerIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
MasterIngressGeneve	Geneve パケット	**udp**	**6081**

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressWorkerGeneve	Geneve パケット	**udp**	**6081**
MasterIngressIpsecIke	IPsec IKE パケット	**udp**	**500**
MasterIngressWorkerIpsecIke	IPsec IKE パケット	**udp**	**500**
MasterIngressIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
MasterIngressWorkerIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
MasterIngressIpsecEsp	IPsec ESP パケット	**50**	**All**
MasterIngressWorkerIpsecEsp	IPsec ESP パケット	**50**	**All**
MasterIngressInternalUDP	内部クラスター通信	**udp**	**9000** - **9999**
MasterIngressWorkerInternalUDP	内部クラスター通信	**udp**	**9000** - **9999**
MasterIngressIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000** - **32767**
MasterIngressWorkerIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000** - **32767**

ワーカーの Ingress

ワーカーマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngressVxlan	Vxlan パケット	**udp**	**4789**
WorkerIngressWorkerVxlan	Vxlan パケット	**udp**	**4789**
WorkerIngressInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngressWorkerInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngressIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
WorkerIngressWorkerIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
WorkerIngressGeneve	Geneve パケット	**udp**	**6081**
WorkerIngressMasterGeneve	Geneve パケット	**udp**	**6081**
WorkerIngressIpsecIke	IPsec IKE パケット	**udp**	**500**
WorkerIngressMasterIpsecIke	IPsec IKE パケット	**udp**	**500**
WorkerIngressIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
WorkerIngressMasterIpsecNat	IPsec NAT-T パケット	**udp**	**4500**

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngressIpsecEsp	IPsec ESP パケット	**50**	**All**
WorkerIngressMasterIpsecEsp	IPsec ESP パケット	**50**	**All**
WorkerIngressInternalUDP	内部クラスター通信	**udp**	**9000 - 9999**
WorkerIngressMasterInternalUDP	内部クラスター通信	**udp**	**9000 - 9999**
WorkerIngressIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000 - 32767**
WorkerIngressMasterIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000 - 32767**

ロールおよびインスタンスプロファイル

マシンには、AWS でのパーミッションを付与する必要があります。提供される CloudFormation テンプレートはマシンに対し、以下の **AWS::IAM::Role** オブジェクトについてのマシンの **Allow** パーミッションを付与し、それぞれのロールセットに **AWS::IAM::InstanceProfile** を指定します。テンプレートを使用しない場合、マシンには以下の広範囲のパーミッションまたは個別のパーミッションを付与することができます。

ロール	結果	アクション	リソース
マスター	**Allow**	**ec2:***	*
	Allow	**elasticloadbalancing:***	*
	Allow	**iam:PassRole**	*
	Allow	**s3:GetObject**	*
ワーカー	**Allow**	**ec2:Describe***	*
ブートストラップ	**Allow**	**ec2:Describe***	*

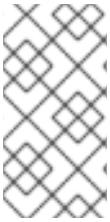
ロール	結果	アクション	リソース
	Allow	**ec2:AttachVolume**	*
	Allow	**ec2:DetachVolume**	*

6.4.2.3.2. クラスターマシン

以下のマシンには **AWS::EC2::Instance** オブジェクトが必要になります。

- ブートストラップマシン。このマシンはインストール時に必要ですが、クラスターのデプロイ後に除去することができます。
- 3 つのコントロールプレーンマシンコントロールプレーンマシンは、コントロールプレーンマシンセットによって管理されません。
- コンピュートマシン。インストール時に 2 つ以上のコンピュートマシン (ワーカーマシンとしても知られる) を作成する必要があります。これらのマシンは、コンピュートマシンセットによって管理されません。

6.4.2.4. IAM ユーザーに必要な AWS パーミッション



注記

ベースクラスターリソースを削除するには、IAM ユーザーが領域 **us-east-1** にアクセス許可 **tag:GetResources** を持っている必要があります。AWS API 要件の一部として、OpenShift Container Platform インストールプログラムはこのリージョンでさまざまなアクションを実行します。

AdministratorAccess ポリシーを、Amazon Web Services (AWS) で作成する IAM ユーザーに割り当てる場合、そのユーザーには必要なパーミッションすべてを付与します。OpenShift Container Platform クラスターのすべてのコンポーネントをデプロイするために、IAM ユーザーに以下のパーミッションが必要になります。

例6.57 インストールに必要な EC2 パーミッション

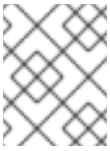
- **ec2:AttachNetworkInterface**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**

- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribePublicIpv4Pools** (**publicIpv4Pool** が **install-config.yaml**で指定されている場合にのみ必要です)
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroupRules**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**

- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:DisassociateAddress** (**publicIpv4Pool** が **install-config.yaml**で指定されている場合にのみ必要)
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

例6.58 インストール時のネットワークリソースの作成に必要なパーミッション

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



注記

既存の Virtual Private Cloud (VPC) を使用する場合、アカウントではネットワークリソースの作成にこれらのパーミッションを必要としません。

例6.59 インストールに必要な Elastic Load Balancing (ELB) のパーミッション

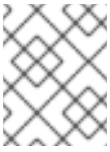
- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**
- **elasticloadbalancing:SetSecurityGroups**

重要

OpenShift Container Platform は、ELB と ELBv2 API サービスの両方を使用してロードバランサーをプロビジョニングします。パーミッションリストには、両方のサービスに必要なパーミッションが表示されます。AWS Web コンソールには、両方のサービスが同じ **elasticloadbalancing** アクション接頭辞を使用しているにもかかわらず、同じアクションを認識しないという既知の問題が存在します。サービスが特定の **elasticloadbalancing** アクションを認識しないという警告は無視できます。

例6.60 インストールに必要な IAM パーミッション

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagInstanceProfile**
- **iam:TagRole**



注記

AWS アカウントにロードバランサーを作成していない場合、IAM ユーザーには **iam:CreateServiceLinkedRole** パーミッションも必要です。

例6.61 インストールに必要な Route 53 パーミッション

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

例6.62 インストールに必要な Amazon Simple Storage Service (S3) パーミッション

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketPolicy**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**

- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

例6.63 クラスター Operator が必要とする S3 パーミッション

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

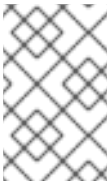
例6.64 ベースクラスターリソースの削除に必要なパーミッション

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeletePlacementGroup**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**

- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

例6.65 ネットワークリソースの削除に必要なパーミッション

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの削除にこれらのパーミッションを必要としません。代わりに、アカウントではネットワークリソースの削除に **tag:UntagResources** パーミッションのみが必要になります。

例6.66 カスタムキー管理サービス (KMS) キーを使用してクラスターをインストールするためのオプションの権限

- **kms:CreateGrant**
- **kms:Decrypt**
- **kms:DescribeKey**
- **kms:Encrypt**
- **kms:GenerateDataKey**
- **kms:GenerateDataKeyWithoutPlainText**
- **kms:ListGrants**

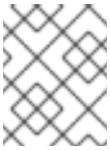
- **kms:RevokeGrant**

例6.67 共有インスタンスロールが割り当てられたクラスターを削除するために必要なパーミッション

- **iam:UntagRole**

例6.68 マニフェストの作成に必要な追加の IAM および S3 パーミッション

- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **s3:AbortMultipartUpload**
- **s3:GetBucketPublicAccessBlock**
- **s3:ListBucket**
- **s3:ListBucketMultipartUploads**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**



注記

クラウドプロバイダーのクレデンシャルをミントモードで管理している場合に、IAM ユーザーには **iam:CreateAccessKey** と **iam:CreateUser** 権限も必要です。

例6.69 インスタンスのオプションのパーミッションおよびインストールのクォータチェック

- **ec2:DescribeInstanceTypeOfferings**
- **servicequotas:ListAWSDefaultServiceQuotas**

例6.70 共有 VPC にクラスターをインストールする場合のクラスター所有者アカウントのオプションの権限

- **sts:AssumeRole**

例6.71 インストールに独自のパブリック IPv4 アドレス(BYOIP)機能を有効にするために必要な権限

- **ec2:DescribePublicIpv4Pools**

- **ec2:DisassociateAddress**

6.4.2.5. AWS Marketplace イメージの取得

AWS Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に AWS を通じてサブスクライブする必要があります。オファーにサブスクライブすると、インストールプログラムがコンピュートノードのデプロイに使用する AMI ID が提供されます。

前提条件

- オファーを購入するための AWS アカウントを持っている。このアカウントは、クラスターのインストールに使用されるアカウントと同じである必要はありません。

手順

1. AWS Marketplace で OpenShift Container Platform サブスクリプションを完了します。

6.4.3. CloudFormation テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、独自に提供するインフラストラクチャーを使用する Amazon Web Services (AWS) にクラスターをインストールできます。

このインフラストラクチャーを作成する 1 つの方法として、提供される CloudFormation テンプレートを使用できます。テンプレートを変更してインフラストラクチャーをカスタマイズしたり、それらに含まれる情報を使用し、所属する会社のポリシーに基づいて AWS オブジェクトを作成したりできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の CloudFormation テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

6.4.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするために AWS アカウントを設定 している。

重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ユーザーによってプロビジョニングされるインフラストラクチャーを準備している。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) を参照してください。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、長期間認証情報を手動で作成および維持 することができます。

6.4.3.2. AWS のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Amazon Web Services (AWS) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

6.4.3.2.1. オプション: 別個の /var パーティションの作成

OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持す

ることができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

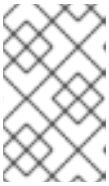
```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
```

```
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

6.4.3.2.2. インストール設定ファイルの作成

インストールプログラムがクラスターをデプロイするために必要なインストール設定ファイルを生成し、カスタマイズします。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャー用の OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- Red Hat が公開している付属の Red Hat Enterprise Linux CoreOS (RHCOS) AMI のある AWS リージョンにクラスターをデプロイしようとしている。カスタム AMI が必要な AWS リージョン (AWS GovCloud リージョンなど) にデプロイする場合は、**install-config.yaml** ファイルを手動で作成する必要があります。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **aws** を選択します。

iii. AWS プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

注記

AWS アクセスキー ID およびシークレットアクセスキーは、インストールホストの現行ユーザーのホームディレクトリーの **~/.aws/credentials** に保存されます。エクスポートされたプロファイルの認証情報がファイルにない場合は、インストールプログラムにより認証情報の入力が求めるプロンプトが出されます。インストールプログラムに指定する認証情報は、ファイルに保存されます。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. 3 ノードクラスターをインストールする場合は、**compute.replicas** パラメーターを **0** に設定して、**install-config.yaml** ファイルを変更します。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細については、「AWS に 3 ノードクラスターをインストールする」を参照してください。

3. オプション: **install-config.yaml** ファイルをバックアップします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- AWS プロファイルおよび認証情報の設定についての詳細は、AWS ドキュメントの Configuration and credential file settings を参照してください。

6.4.3.2.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

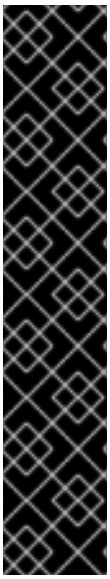
注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.4.3.2.4. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。

- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

重要

ユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールするときに **MachineAPI** 機能を無効にした場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除する必要があります。そうしないと、クラスターのインストールに失敗します。

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

   これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

   Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

6.4.3.3. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Amazon Web Services (AWS) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な AWS リソースを見つけるためにも使用されます。提供される CloudFormation テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

6.4.3.4. AWS での VPC の作成

OpenShift Container Platform クラスターで使用する Virtual Private Cloud (VPC) を Amazon Web Services (AWS) で作成する必要があります。VPN およびルートテーブルを含む、各種要件を満たすように VPC をカスタマイズできます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、VPC を表す AWS リソースのスタックを作成できます。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。

- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "VpcCidr", (1)
    "ParameterValue": "10.0.0.0/16" (2)
  },
  {
    "ParameterKey": "AvailabilityZoneCount", (3)
    "ParameterValue": "1" (4)
  },
  {
    "ParameterKey": "SubnetBits", (5)
    "ParameterValue": "12" (6)
  }
]
```

(1) VPC の CIDR ブロック。

(2) **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。

(3) VPC をデプロイするアベイラビリティーゾーンの数。

(4) **1** から **3** の間の整数を指定します。

(5) 各アベイラビリティーゾーン内の各サブネットのサイズ。

(6) **5** から **13** の間の整数を指定します。 ここで、**5** は **/27** であり、 **13** は **/19** です。

2. このトピックの**VPC の CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

3. CloudFormation テンプレートを起動し、VPC を表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> (1)
    --template-body file://<template>.yaml (2)
    --parameters file://<parameters>.json (3)
```

(1) **<name>** は **cluster-vpc** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-vpc/dbedae40-2fd3-11eb-820e-12a48460849f
```

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

VpcId	VPC の ID。
PublicSubnetIds	新規パブリックサブネットの ID。
PrivateSubnetIds	新規プライベートサブネットの ID。

6.4.3.4.1. VPC の CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例6.72 VPC の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
```

```
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 =
/19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
```

```
    - Fn::GetAZs: !Ref "AWS::Region"
PublicSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
InternetGateway:
  Type: "AWS::EC2::InternetGateway"
GatewayToInternet:
  Type: "AWS::EC2::VPCGatewayAttachment"
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway
PublicRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PublicRoute:
  Type: "AWS::EC2::Route"
  DependsOn: GatewayToInternet
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation3:
  Condition: DoAz3
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet3
    RouteTableId: !Ref PublicRouteTable
PrivateSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 0
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
```

```
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable2
  NAT2:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz2
    Properties:
      AllocationId:
        "Fn::GetAtt":
```

```
      - EIP2
      - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP3
      - AllocationId
    SubnetId: !Ref PublicSubnet3
EIP3:
  Type: "AWS::EC2::EIP"
  Condition: DoAz3
  Properties:
    Domain: vpc
Route3:
  Type: "AWS::EC2::Route"
  Condition: DoAz3
```

```
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
  PrivateRouteTableIds:
    Description: Private Route table IDs
    Value:
```

```
      !Join [
        ",",
        [
          !Join ["=", [
            !Select [0, "Fn::GetAZs": !Ref "AWS::Region"],
            !Ref PrivateRouteTable
          ]],
          !If [DoAz2,
              !Join ["=", [!Select [1, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable2]],
              !Ref "AWS::NoValue"
          ],
          !If [DoAz3,
              !Join ["=", [!Select [2, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable3]],
              !Ref "AWS::NoValue"
          ]
        ]
      ]
```

関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

6.4.3.5. AWS でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用できるネットワークおよび負荷分散 (classic または network) を Amazon Web Services (AWS) で設定する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform クラスターに必要なネットワークおよび負荷分散コンポーネントを表します。テンプレートは、ホストゾーンおよびサブネットタグも作成します。

単一 Virtual Private Cloud 内でテンプレートを複数回実行することができます。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。

手順

1. クラスターの **install-config.yaml** ファイルに指定した Route 53 ベースドメインのホストゾーン ID を取得します。以下のコマンドを実行して、ホストゾーンの詳細を取得できます。

```
$ aws route53 list-hosted-zones-by-name --dns-name <route53_domain> 1
```

1 **<route53_domain>** について、クラスターの **install-config.yaml** ファイルを生成した時に作成した Route 53 ベースドメインを指定します。

出力例

```
mycluster.example.com. False 100
HOSTEDZONES 65F8F38E-2268-B835-E15C-AB55336FCBFA
/hostedzone/Z21IXYZABCZ2A4 mycluster.example.com. 10
```

この出力例では、ホストゾーン ID は **Z21IXYZ3-2Z2A4** です。

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "ClusterName", 1
    "ParameterValue": "mycluster" 2
  },
  {
    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
    "ParameterValue": "<random_string>" 6
  },
  {
    "ParameterKey": "HostedZoneName", 7
    "ParameterValue": "example.com" 8
  },
  {
    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]
```

1 ホスト名などに使用する短いクラスター名。

2 クラスターの **install-config.yaml** ファイルを生成した時に使用したクラスター名を指定します。

3 クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。

4 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

5 ターゲットの登録に使用する Route 53 パブリックトゾーン ID。

6 **Z21IXYZABCZ2A4** に類する形式の Route 53 パブリックゾーン ID を指定します。この値は AWS コンソールから取得できます。

7 ターゲットの登録に使用する Route 53 ゾーン。

8 クラスターの **install-config.yaml** ファイルを生成した時に使用した Route 53 ベースドメインを指定します。AWS コンソールに表示される末尾のピリド (.) は含めないでください。

9 VPC 用に作成したパブリックサブネット。

10 VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。

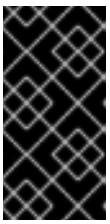
11 VPC 用に作成したプライベートサブネット。

12 VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。

13 クラスター用に作成した VPC。

14 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

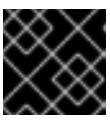
3. このトピックの**ネットワークおよびロードバランサーの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。



重要

AWS government またはシークレットリージョンにクラスターをデプロイする場合は、CloudFormation テンプレートの **InternalApiServerRecord** を更新して、**CNAME** レコードを使用する必要があります。**ALIAS** タイプのレコードは、AWS 政府リージョンではサポートされません。

4. CloudFormation テンプレートを起動し、ネットワークおよび負荷分散コンポーネントを提供する AWS リソースのスタックを作成します。



重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

```
  --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>** は **cluster-dns** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。
4. 提供されるテンプレートは一部の **AWS::IAM::Role** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-dns/cd3e5de0-2fd4-11eb-5cf0-12be5c33a183
```

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

PrivateHostedZoneId	プライベート DNS のホストゾーン ID。
ExternalApiLoadBalancerName	外部 API ロードバランサーのフルネーム。
InternalApiLoadBalancerName	内部 API ロードバランサーのフルネーム。
ApiServerDnsName	API サーバーの完全ホスト名。
RegisterNlbIpTargetsLambda	これらのロードバランサーの登録/登録解除に役立つ Lambda ARN。
ExternalApiTargetGroupArn	外部 API ターゲットグループの ARN。

InternalApiTargetGroupArn	内部 API ターゲットグループの ARN。
InternalServiceTargetGroupArn	内部サービスターゲットグループの ARN。

6.4.3.5.1. ネットワークおよびロードバランサーの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例6.73 ネットワークおよびロードバランサーの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
```

```
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
        default: "Public Subnets"
      PrivateSubnets:
        default: "Private Subnets"
      HostedZoneName:
        default: "Public Hosted Zone Name"
      HostedZoneId:
        default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network
```

```
IntDns:
  Type: "AWS::Route53::HostedZone"
  Properties:
    HostedZoneConfig:
      Comment: "Managed by CloudFormation"
    Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
    HostedZoneTags:
    - Key: Name
      Value: !Join ["-", [!Ref InfrastructureName, "int"]]
    - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
      Value: "owned"
    VPCs:
    - VPCId: !Ref VpcId
      VPCRegion: !Ref "AWS::Region"

ExternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref HostedZoneId
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt ExtApiElb.DNSName

InternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref IntDns
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
    - Name:
        !Join [
          ".",
          ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
```

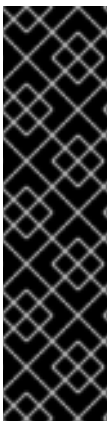
```
  ExternalApiListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: ExternalApiTargetGroup
      LoadBalancerArn:
        Ref: ExtApiElb
      Port: 6443
      Protocol: TCP

  ExternalApiTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/readyz"
      HealthCheckPort: 6443
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 6443
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  InternalApiListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalApiTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 6443
      Protocol: TCP

  InternalApiTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/readyz"
      HealthCheckPort: 6443
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 6443
      Protocol: TCP
      TargetType: ip
      VpcId:
```

```
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

InternalServiceInternalListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalServiceTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 22623
    Protocol: TCP

InternalServiceTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/healthz"
    HealthCheckPort: 22623
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 22623
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

RegisterTargetLambdaIamRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
          Service:
          - "lambda.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Path: "/"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
```

```
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
            Resource: !Ref InternalApiTargetGroup
          - Effect: "Allow"
            Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
            Resource: !Ref InternalServiceTargetGroup
          - Effect: "Allow"
            Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
            Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.8"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
```

```
          Service:
          - "lambda.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Path: "/"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
            [
              "ec2:DeleteTags",
              "ec2:CreateTags"
            ]
          Resource: "arn:aws:ec2:*:*:subnet/*"
        - Effect: "Allow"
          Action:
            [
              "ec2:DescribeSubnets",
              "ec2:DescribeTags"
            ]
          Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.8"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
```

```
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
    Value: !Ref ExternalApiTargetGroup
  InternalApiTargetGroupArn:
    Description: ARN of the internal API target group.
    Value: !Ref InternalApiTargetGroup
  InternalServiceTargetGroupArn:
    Description: ARN of the internal service target group.
    Value: !Ref InternalServiceTargetGroup
```



重要

クラスターを AWS government またはシークレットリージョンにデプロイする場合は、**InternalApiServerRecord** を更新し、**CNAME** レコードを使用する必要があります。**ALIAS** タイプのレコードは、AWS 政府リージョンではサポートされません。以下に例を示します。

```
Type: CNAME
TTL: 10
ResourceRecords:
- !GetAtt IntApiElb.DNSName
```

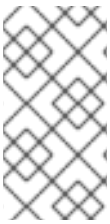
関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。
- AWS Route 53 コンソール に移動して、ホストゾーンについての詳細を表示できます。
- パブリックホストゾーンのリスト表示についての詳細は、AWS ドキュメントの Listing public hosted zones を参照してください。

6.4.3.6. AWS でのセキュリティーグループおよびロールの作成

OpenShift Container Platform クラスターで使用するセキュリティーグループおよびロールを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform クラスターに必要なセキュリティーグループおよびロールを表します。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
```

```
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. VPC の CIDR ブロック。
4. **x.x.x.x/16-24** の形式で定義した VPC に使用した CIDR ブロックパラメーターを指定します。
5. VPC 用に作成したプライベートサブネット。
6. VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。
7. クラスター用に作成した VPC。
8. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

2. このトピックの**セキュリティーオブジェクトの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なセキュリティーグループおよびロールについて記述しています。
3. CloudFormation テンプレートを起動し、セキュリティーグループおよびロールを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>** は **cluster-secs** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。
4. 提供されるテンプレートは一部の **AWS::IAM::Role** および **AWS::IAM::InstanceProfile** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-sec/03bd4210-2ed7-11eb-6d7a-13fc0b61e9db
```

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

MasterSecurityGroupId	マスターセキュリティーグループ ID
WorkerSecurityGroupId	ワーカーセキュリティーグループ ID
MasterInstanceProfile	マスター IAM インスタンスプロファイル
WorkerInstanceProfile	ワーカー IAM インスタンスプロファイル

6.4.3.6.1. セキュリティーオブジェクトの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なセキュリティーオブジェクトをデプロイすることができます。

例6.74 セキュリティーオブジェクトの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
```

```
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        ToPort: 6443
        FromPort: 6443
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22623
        ToPort: 22623
        CidrIp: !Ref VpcCidr
```

```
    VpcId: !Ref VpcId

WorkerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Cluster Worker Security Group
    SecurityGroupIngress:
    - IpProtocol: icmp
      FromPort: 0
      ToPort: 0
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref VpcCidr
    VpcId: !Ref VpcId

MasterIngressEtcd:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: etcd
    FromPort: 2379
    ToPort: 2380
    IpProtocol: tcp

MasterIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressWorkerVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp
```

```
  MasterIngressWorkerGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  MasterIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  MasterIngressIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  MasterIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  MasterIngressWorkerIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  MasterIngressWorkerIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
```

```
    ToPort: 4500
    IpProtocol: udp

  MasterIngressWorkerIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  MasterIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressWorkerInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressWorkerInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
```

```
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressWorkerKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressWorkerIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

MasterIngressWorkerIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressVxlan:
```

```
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressMasterVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

WorkerIngressMasterGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

WorkerIngressIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

WorkerIngressIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
```

```
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  WorkerIngressMasterIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  WorkerIngressMasterIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressMasterIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  WorkerIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressMasterInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
```

```
    ToPort: 9999
    IpProtocol: tcp

  WorkerIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressMasterInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes secure kubelet port
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal Kubernetes communication
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressMasterIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
```

```
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

  WorkerIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressMasterIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:AttachVolume"
            - "ec2:AuthorizeSecurityGroupIngress"
            - "ec2:CreateSecurityGroup"
            - "ec2:CreateTags"
            - "ec2:CreateVolume"
            - "ec2:DeleteSecurityGroup"
            - "ec2:DeleteVolume"
            - "ec2:Describe*"
            - "ec2:DetachVolume"
            - "ec2:ModifyInstanceAttribute"
```

```
          - "ec2:ModifyVolume"
          - "ec2:RevokeSecurityGroupIngress"
          - "elasticloadbalancing:AddTags"
          - "elasticloadbalancing:AttachLoadBalancerToSubnets"
          - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
          - "elasticloadbalancing:CreateListener"
          - "elasticloadbalancing:CreateLoadBalancer"
          - "elasticloadbalancing:CreateLoadBalancerPolicy"
          - "elasticloadbalancing:CreateLoadBalancerListeners"
          - "elasticloadbalancing:CreateTargetGroup"
          - "elasticloadbalancing:ConfigureHealthCheck"
          - "elasticloadbalancing:DeleteListener"
          - "elasticloadbalancing:DeleteLoadBalancer"
          - "elasticloadbalancing:DeleteLoadBalancerListeners"
          - "elasticloadbalancing:DeleteTargetGroup"
          - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
          - "elasticloadbalancing:DeregisterTargets"
          - "elasticloadbalancing:Describe*"
          - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
          - "elasticloadbalancing:ModifyListener"
          - "elasticloadbalancing:ModifyLoadBalancerAttributes"
          - "elasticloadbalancing:ModifyTargetGroup"
          - "elasticloadbalancing:ModifyTargetGroupAttributes"
          - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
          - "elasticloadbalancing:RegisterTargets"
          - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
          - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
          - "kms:DescribeKey"
          Resource: "*"

MasterInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Roles:
    - Ref: "MasterIamRole"

WorkerIamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
          Service:
          - "ec2.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
          - "ec2:DescribeInstances"
```

```
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

  MasterInstanceProfile:
    Description: Master IAM Instance Profile
    Value: !Ref MasterInstanceProfile

  WorkerInstanceProfile:
    Description: Worker IAM Instance Profile
    Value: !Ref WorkerInstanceProfile
```

関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

6.4.3.7. ストリームメタデータを使用した RHCOS AMI へのアクセス

OpenShift Container Platform では、**ストリームメタデータ** は、JSON 形式で RHCOS に関する標準化されたメタデータを 提供し、メタデータをクラスターに挿入します。ストリームメタデータは、複数のアーキテクチャーをサポートする安定した形式で、自動化を維持するための自己文書化が意図されています。

openshift-install の **coreos print-stream-json** サブコマンドを使用して、ストリームメタデータ形式のブートイメージに関する情報にアクセスできます。このコマンドは、スクリプト可能でマシン読み取り可能な形式でストリームメタデータを出力する方法を提供します。

ユーザーによってプロビジョニングされるインストールの場合、**openshift-install** バイナリーには、AWS AMI などの OpenShift Container Platform での使用がテストされている RHCOS ブートイメージのバージョンへの参照が含まれます。

手順

ストリームメタデータを解析するには、以下のいずれかの方法を使用します。

- Go プログラムから、https://github.com/coreos/stream-metadata-go の公式の **stream-metadata-go** ライブラリーを使用します。ライブラリーでサンプルコードを確認することもできます。

- Python や Ruby などの別のプログラミング言語から、お好みのプログラミング言語の JSON ライブラリーを使用します。
- **jq** などの JSON データを処理するコマンドラインユーティリティーから、以下のコマンドを実行します。
  - **us-west-1** などの AWS リージョンの現在の **x86_64** または **aarch64** AMI を出力します。

    x86_64 の場合

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.x86_64.images.aws.regions["us-west-1"].image'
    ```

    出力例

    ```
    ami-0d3e625f84626bbda
    ```

    aarch64 の場合

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.aarch64.images.aws.regions["us-west-1"].image'
    ```

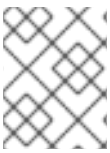
    出力例

    ```
    ami-0af1d3b7fa5be2131
    ```

    このコマンドの出力は、指定されたアーキテクチャーと **us-west-1** リージョンの AWS AMI ID です。AMI はクラスターと同じリージョンに属する必要があります。

6.4.3.8. AWS インフラストラクチャーの RHCOS AMI

Red Hat は、OpenShift Container Platform ノードに手動で指定できる、さまざまな AWS リージョンおよびインスタンスアーキテクチャーに有効な Red Hat Enterprise Linux CoreOS(RHCOS) AMI を提供します。



注記

また、独自の AMI をインポートすることで、RHCOS AMI がパブリッシュされていないリージョンにインストールすることもできます。

表6.23 x86_64 RHCOS AMIs

AWS ゾーン	AWS AMI
af-south-1	**ami-018de6b1be470b7e6**
ap-east-1	**ami-092f09a4c8aacf56a**
ap-northeast-1	**ami-001c9fc85b77505f4**
ap-northeast-2	**ami-0a5d7f1966d88fba3**

AWS ゾーン	AWS AMI
ap-northeast-3	ami-085a2726ceb4f5eeb
ap-south-1	ami-03f2c19071fb6eab3
ap-south-2	ami-0c82522890979d94b
ap-southeast-1	ami-06e5b9d16ead6c55c
ap-southeast-2	ami-0ccd429129fbf6bc0
ap-southeast-3	ami-096f9b4d41116d95c
ap-southeast-4	ami-0b511ab55f9f7d052
ca-central-1	ami-0e357287c651be1f2
ca-west-1	ami-0b1692e5776740901
eu-central-1	ami-08b13fb388ba5610d
eu-central-2	ami-04777298b55045ea9
eu-north-1	ami-05421993a4ee567b9
eu-south-1	ami-00959341869d34746
eu-south-2	ami-0a745b610522a87cc
eu-west-1	ami-0e48f85ec57bd7baa
eu-west-2	ami-0c015b1387acddc5e
eu-west-3	ami-01e466af77a95b93d
il-central-1	ami-0a1b706793b54d087
me-central-1	ami-09d58e8b2099ae5bc
me-south-1	ami-0f9c259bc4346599c
sa-east-1	ami-06277517d966881a3
us-east-1	ami-05483066c3caaccf5
us-east-2	ami-0c704ce516493a479

AWS ゾーン	AWS AMI
us-gov-east-1	ami-0695b2cbdd1ec4d48
us-gov-west-1	ami-004404a0eaa427b39
us-west-1	ami-0aaa56637063be772
us-west-2	ami-0870c7a3d5ef4d2b3

表6.24 aarch64 RHCOS AMI

AWS ゾーン	AWS AMI
af-south-1	ami-0d96d447d3df14f4e
ap-east-1	ami-010236402dfba1717
ap-northeast-1	ami-08feeb6d9068bb12e
ap-northeast-2	ami-0772082c119147205
ap-northeast-3	ami-05bcbb13493d0516f
ap-south-1	ami-0034a539e8adcb817
ap-south-2	ami-0fc56b7c53c38ff70
ap-southeast-1	ami-0b50a0e92a58f3ad8
ap-southeast-2	ami-0697a9174ae206182
ap-southeast-3	ami-05ae309319a7ad504
ap-southeast-4	ami-00500414843baa657
ca-central-1	ami-05e76bf99d3b0d4c8
ca-west-1	ami-099fc953c221ec590
eu-central-1	ami-0d2e2698884020b71
eu-central-2	ami-0a96ba21ddee01719
eu-north-1	ami-0ab8a1070d0b1d9a5

AWS ゾーン	AWS AMI
eu-south-1	**ami-0d0465299a8b196c1**
eu-south-2	**ami-07d5aa3c6d468c738**
eu-west-1	**ami-08e7d209f26c09c80**
eu-west-2	**ami-0c8336d4e2c1f13cb**
eu-west-3	**ami-085d804b98acf0648**
il-central-1	**ami-02ce030e36aeb7643**
me-central-1	**ami-0dea3ac466040b77f**
me-south-1	**ami-02ef2a46887b9786d**
sa-east-1	**ami-0d19817d31b2399f9**
us-east-1	**ami-04859c888566a2c2f**
us-east-2	**ami-02f7e4a5dc66361bb**
us-gov-east-1	**ami-067ef782980ea96ad**
us-gov-west-1	**ami-0ce67540c1bb2319d**
us-west-1	**ami-0a09c5d2f287718a3**
us-west-2	**ami-06b94e64e595f6cee**

6.4.3.8.1. 公開済み RHCOS AMI のない AWS リージョン

Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) または AWS software development kit (SDK) のネイティブサポートなしに、OpenShift Container Platform クラスターを Amazon Web Services (AWS) リージョンにデプロイできます。パブリッシュ済みの AMI が AWS リージョンで利用できない場合は、クラスターをインストールする前にカスタム AMI をアップロードできます。

AWS SDK によってサポートされないリージョンにデプロイしている場合で、カスタム AMI を指定しない場合、インストールプログラムは **us-east-1** AMI をユーザーアカウントに自動的にコピーします。次にインストールプログラムは、デフォルトまたはユーザー指定の Key Management Service (KMS) キーを使用して、暗号化された EBS ボリュームでコントロールプレーンマシンを作成します。これにより、AMI は、パブリッシュ済みの RHCOS AMI と同じプロセスワークフローを実施することができます。

RHCOS AMI のネイティブサポートのないリージョンは、パブリッシュされたクラスターの作成

RHCOS AMI のネイティブサポートのないリージョンはバブリッシュされないため、クラスターの作成時にターミナルから選択することはできません。ただし、**install-config.yaml** ファイルでカスタム AMI を設定して、このリージョンにインストールすることができます。

6.4.3.8.2. AWS でのカスタム RHCOS AMI のアップロード

カスタム Amazon Web Services (AWS) リージョンにデプロイする場合、そのリージョンに属するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) をアップロードする必要があります。

前提条件

- AWS アカウントを設定している。
- 必要な IAM サービスロール で、Amazon S3 バケットを作成している。
- RHCOS VMDK ファイルを Amazon S3 にアップロードしている。RHCOS VMDK ファイルは、インストールする OpenShift Container Platform のバージョンと同じか、それ以下のバージョンである必要があります。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer を参照してください。

手順

1. AWS プロファイルを環境変数としてエクスポートします。

```
$ export AWS_PROFILE=<aws_profile> 1
```

2. カスタム AMI に関連付けるリージョンを環境変数としてエクスポートします。

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

3. 環境変数として Amazon S3 にアップロードした RHCOS のバージョンをエクスポートします。

```
$ export RHCOS_VERSION=<version> 1
```

1 1 1 **4.16.0** などの RHCOS VMDK バージョン。

4. Amazon S3 バケット名を環境変数としてエクスポートします。

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. **containers.json** ファイルを作成し、RHCOS VMDK ファイルを定義します。

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
```

```
  }
}
EOF
```

6. RHCOS ディスクを Amazon EBS スナップショットとしてインポートします。

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1 **rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64** などの RHCOS ディスクがインポートされていることの説明。

2 RHCOS ディスクを説明する JSON ファイルへのファイルパス。JSON ファイルには、Amazon S3 バケット名とキーが含まれている必要があります。

7. イメージインポートのステータスを確認します。

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

出力例

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

SnapshotId をコピーして、イメージを登録します。

8. RHCOS スナップショットからカスタム RHCOS AMI を作成します。

```
$ aws ec2 register-image \
  --region ${AWS_DEFAULT_REGION} \
  --architecture x86_64 \ 1
  --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
  --ena-support \
  --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
  --virtualization-type hvm \
```

```
    --root-device-name '/dev/xvda' \
    --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

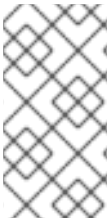
1. **x86_64**、**aarch64**、**s390x**、または **ppc64le** などの RHCOS VMDK アーキテクチャータイプ。
2. インポートされたスナップショットの **Description**。
3. RHCOS AMI の名前。
4. インポートされたスナップショットからの **SnapshotID**。

これらの API の詳細は、AWS ドキュメントの Importing a Disk as a Snapshot Using VM Import/Export および Creating a Linux AMI from a snapshot を参照してください。

6.4.3.9. AWS でのブートストラップノードの作成

OpenShift Container Platform クラスターの初期化で使用するブートストラップノードを Amazon Web Services (AWS) で作成する必要があります。これは、以下の方法で行います。

- **bootstrap.ign** Ignition 設定ファイルをクラスターに送るための場所を指定。このファイルはインストールディレクトリーに置かれます。提供される CloudFormation テンプレートでは、クラスターの Ignition 設定ファイルは S3 バケットから送られることを前提としています。このファイルを別の場所から送ることを選択する場合は、テンプレートを変更する必要があります。
- 提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform インストールに必要なブートストラップノードを表します。



注記

提供される CloudFormation テンプレートを使用してブートストラップノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。

手順

1. 以下のコマンドを実行してバケットを作成します。

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra** はバケット名です。**install-config.yaml** ファイルを作成する際に、**<cluster-name>** をクラスターに指定された名前に置き換えます。

以下の場合は、**s3://** スキーマではなく、S3 バケットに事前に署名された URL を使用する必要があります。

- AWS SDK とは異なるエンドポイントを持つリージョンへのデプロイ。
- プロキシーをデプロイする。
- カスタムエンドポイントを指定します。

2. 以下のコマンドを実行して **bootstrap.ign** Ignition 設定ファイルをバケットにアップロードします。

```
$ aws s3 cp <installation_directory>/bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

3. 以下のコマンドを実行して、ファイルがアップロードされていることを確認します。

```
$ aws s3 ls s3://<cluster-name>-infra/
```

出力例

```
2019-04-03 16:15:16     314878 bootstrap.ign
```

注記

ブートストラップ Ignition 設定ファイルには、X.509 キーのようなシークレットが含まれません。以下の手順では、S3 バケットの基本的なセキュリティーを提供します。追加のセキュリティーを提供するには、OpenShift IAM ユーザーなどの特定のユーザーのみがバケットに含まれるオブジェクトにアクセスできるように S3 バケットポリシーを有効にできます。S3 を完全に回避し、ブートストラップマシンが到達できるアドレスからブートストラップ Ignition 設定ファイルを送ることができます。

4. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
```

```
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 9
    "ParameterValue": "sg-<random_string>" 10
  },
  {
    "ParameterKey": "VpcId", 11
    "ParameterValue": "vpc-<random_string>" 12
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", 13
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
  },
  {
    "ParameterKey": "AutoRegisterELB", 15
    "ParameterValue": "yes" 16
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
    "ParameterValue": "arn:aws:lambda:<aws_region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 19
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

3. 選択したアーキテクチャーに基づいてブートストラップノードに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. 有効な **AWS::EC2::Image::Id** 値を指定します。
5. ブートストラップノードへの SSH アクセスを許可する CIDR ブロック。
6. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
7. ブートストラップを起動するために VPC に関連付けられるパブリックサブネット。
8. VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。
9. マスターセキュリティーグループ ID (一時ルールの登録用)。
10. セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。
11. 作成されたリソースが属する VPC。
12. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。
13. ブートストラップの Ignition 設定ファイルをフェッチする場所。
14. **s3://<bucket_name>/bootstrap.ign** の形式で S3 バケットおよびファイル名を指定します。
15. ネットワークロードバランサー (NLB) を登録するかどうか。
16. **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。
17. NLB IP ターゲット登録 lambda グループの ARN。
18. DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。
19. 外部 API ロードバランサーのターゲットグループの ARN。
20. DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。
21. 内部 API ロードバランサーのターゲットグループの ARN。
22. DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。
23. 内部サービスバランサーのターゲットグループの ARN。
24. DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

5. このトピックの**ブートストラップマシンの CloudFormation テンプレート**セクションからテン

5. このトピックの**ブートストラップマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。

6. オプション： プロキシーを使用してクラスターをデプロイする場合は、テンプレートの ignition を更新して **ignition.config.proxy** フィールドを追加する必要があります。さらに、Amazon EC2、Elastic Load Balancing、および S3 VPC エンドポイントを VPC に追加している場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。

7. CloudFormation テンプレートを起動し、ブートストラップノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>** は **cluster-bootstrap** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4. 提供されるテンプレートは一部の **AWS::IAM::Role** および **AWS::IAM::InstanceProfile** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-bootstrap/12944486-2add-11eb-9dee-12dace8e3a83
```

8. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

BootstrapInstanceId	ブートストラップインスタンス ID。
BootstrapPublicIp	ブートストラップノードのパブリック IP アドレス。

Bootstrap PrivateIp	ブートストラップノードのプライベート IP アドレス。

6.4.3.9.1. ブートストラップマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイできます。

例6.75 ブートストラップマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
```

```
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String
  BootstrapInstanceType:
    Description: Instance type for the bootstrap EC2 instance
    Default: "i3.large"
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      AllowedBootstrapSshCidr:
        default: "Allowed SSH Source"
      PublicSubnet:
        default: "Public Subnet"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Bootstrap Ignition Source"
```

```
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  BootstrapIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action: "ec2:Describe*"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:AttachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:DetachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "s3:GetObject"
            Resource: "*"

  BootstrapInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Path: "/"
      Roles:
      - Ref: "BootstrapIamRole"

  BootstrapSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Bootstrap Security Group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref AllowedBootstrapSshCidr
```

```
      - IpProtocol: tcp
        ToPort: 19531
        FromPort: 19531
        CidrIp: 0.0.0.0/0
      VpcId: !Ref VpcId

  BootstrapInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      IamInstanceProfile: !Ref BootstrapInstanceProfile
      InstanceType: !Ref BootstrapInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "true"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "BootstrapSecurityGroup"
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "PublicSubnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"replace":{"source":"${S3Loc}"}},"version":"3.1.0"}}'
        - {
          S3Loc: !Ref BootstrapIgnitionLocation
        }

  RegisterBootstrapApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

  BootstrapPublicIp:
```

```
  Description: The bootstrap node public IP address.
  Value: !GetAtt BootstrapInstance.PublicIp

BootstrapPrivateIp:
  Description: The bootstrap node private IP address.
  Value: !GetAtt BootstrapInstance.PrivateIp
```

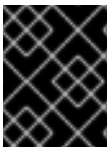
関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。
- AWS ゾーンの Red Hat Enterprise Linux CoreOS (RHCOS) AMI についての詳細は、 AWS インフラストラクチャーの RHCOS AMI を参照してください。

6.4.3.10. AWS でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、コントロールプレーンノードを表す AWS リソースのスタックを作成できます。



重要

CloudFormation テンプレートは、3 つのコントロールプレーンノードを表すスタックを作成します。

注記

提供される CloudFormation テンプレートを使用してコントロールプレーンノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
20
  },
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", 23
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
  },
  {
```

```
    "ParameterKey": "MasterInstanceType", 25
    "ParameterValue": "" 26
  },
  {
    "ParameterKey": "AutoRegisterELB", 27
    "ParameterValue": "yes" 28
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 29
    "ParameterValue": "arn:aws:lambda:<aws_region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 30
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 31
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. 選択したアーキテクチャーに基づいてコントロールプレーンマシンに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. DNS etcd 登録を実行するかどうか。
6. **yes** または **no** を指定します。**yes** を指定する場合、ホストゾーンの情報を指定する必要があります。
7. etcd ターゲットの登録に使用する Route 53 プライベートゾーン ID。
8. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateHostedZoneId** 値を指定します。
9. ターゲットの登録に使用する Route 53 ゾーン。
10. **<cluster_name>.<domain_name>** を指定します。ここで、**<domain_name>** はクラスターの **install-config.yaml** ファイルの生成時に使用した Route 53 ベースドメインです。AWS コンソールに表示される末尾のピリド (.) は含めないでください。

11 13 15 コントロールプレーンマシンの起動に使用するサブネット (プライベートが望ましい)。

12 14 16 DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。

17 コントロールプレーンノードに関連付けるマスターセキュリティーグループ ID。

18 セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。

19 コントロールプレーンの Ignition 設定ファイルをフェッチする場所。

20 生成される Ignition 設定ファイルの場所を指定します (**https://api-int.<cluster_name>.<domain_name>:22623/config/master**)。

21 使用する base64 でエンコードされた認証局の文字列。

22 インストールディレクトリーにある **master.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

23 コントロールプレーンノードに関連付ける IAM プロファイル。

24 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **MasterInstanceProfile** パラメーターの値を指定します。

25 選択したアーキテクチャーに基づいてコントロールプレーンマシンに使用する AWS インスタンスのタイプ。

26 インスタンスタイプの値は、コントロールプレーンマシンの最小リソース要件に対応します。たとえば、**m6i.xlarge** は AMD64 のタイプであり、**m6g.xlarge** は、ARM64 のタイプです。

27 ネットワークロードバランサー (NLB) を登録するかどうか。

28 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

29 NLB IP ターゲット登録 lambda グループの ARN。

30 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

31 外部 API ロードバランサーのターゲットグループの ARN。

32 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

33 内部 API ロードバランサーのターゲットグループの ARN。

34 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

35 内部サービスバランサーのターゲットグループの ARN。

36 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リー

2. このトピックの**コントロールプレーンマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。

3. **m5** インスタンスタイプを **MasterInstanceType** の値として指定している場合、そのインスタンスタイプを CloudFormation テンプレートの **MasterInstanceType.AllowedValues** パラメーターに追加します。

4. CloudFormation テンプレートを起動し、コントロールプレーンノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

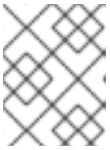
1 **<name>** は **cluster-control-plane** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-control-plane/21c7e2b0-2ee2-11eb-c6f6-0aa34627df4b
```



注記

CloudFormation テンプレートは、3 つのコントロールプレーンノードを表すスタックを作成します。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

6.4.3.10.1. コントロールプレーンマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例6.76 コントロールプレーンマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: ""
    Description: unused
    Type: String
  PrivateHostedZoneId:
    Default: ""
    Description: unused
    Type: String
  PrivateHostedZoneName:
    Default: ""
    Description: unused
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m5.xlarge
    Type: String
```

```
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - MasterInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - MasterSecurityGroupId
      - MasterInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - Master0Subnet
      - Master1Subnet
      - Master2Subnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
```

```
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      Master0Subnet:
        default: "Master-0 Subnet"
      Master1Subnet:
        default: "Master-1 Subnet"
      Master2Subnet:
        default: "Master-2 Subnet"
      MasterInstanceType:
        default: "Master Instance Type"
      MasterInstanceProfileName:
        default: "Master Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Master Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
```

```
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"
```

```
  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
```

```
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref ExternalApiTargetGroupArn
        TargetIp: !GetAtt Master2.PrivateIp

    RegisterMaster2InternalApiTarget:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref InternalApiTargetGroupArn
        TargetIp: !GetAtt Master2.PrivateIp

    RegisterMaster2InternalServiceTarget:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref InternalServiceTargetGroupArn
        TargetIp: !GetAtt Master2.PrivateIp

  Outputs:
    PrivateIPs:
      Description: The control-plane node private IP addresses.
      Value:
        !Join [
          ",",
          [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
        ]
```

関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

6.4.3.11. AWS でのワーカーノードの作成

クラスターで使用するワーカーノードを Amazon Web Services (AWS) で作成できます。

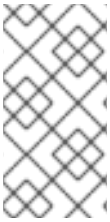
注記

3 ノードクラスターをインストールする場合は、この手順をスキップしてください。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコントロールプレーンマシンで設定されます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、ワーカーノードを表す AWS リソースのスタックを作成できます。

重要

CloudFormation テンプレートは、1 つのワーカーノードを表すスタックを作成します。それぞれのワーカーノードにスタックを作成する必要があります。



注記

提供される CloudFormation テンプレートを使用してワーカーノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。
- コントロールプレーンマシンを作成している。

手順

1. CloudFormation テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
  },
  {
    "ParameterKey": "CertificateAuthorities", 11
```

```
    "ParameterValue": "" (12)
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", (13)
    "ParameterValue": "" (14)
  },
  {
    "ParameterKey": "WorkerInstanceType", (15)
    "ParameterValue": "" (16)
  }
]
```

(1) クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。

(2) 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

(3) 選択したアーキテクチャーに基づいてワーカーノードに使用する現在の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。

(4) **AWS::EC2::Image::Id** 値を指定します。

(5) ワーカーノードを起動するためのサブネット (プライベートであることが望ましい)。

(6) DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。

(7) ワーカーノードに関連付けるワーカーセキュリティーグループ ID。

(8) セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WorkerSecurityGroupId** 値を指定します。

(9) ブートストラップ Ignition 設定ファイルを取得する場所。

(10) 生成される Ignition 設定の場所を指定します。 **https://api-int.<cluster_name>.<domain_name>:22623/config/worker**

(11) 使用する base64 でエンコードされた認証局の文字列。

(12) インストールディレクトリーにある **worker.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

(13) ワーカーロールに関連付ける IAM プロファイル。

(14) セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WokerInstanceProfile** パラメーターの値を指定します。

(15) 選択したアーキテクチャーに基づいてコンピュートマシンに使用する AWS インスタンスのタイプ。

(16) インスタンスタイプの値は、コンピュートマシンの最小リソース要件に対応します。たとえば、**m6i.large** は AMD64 のタイプであり、 **m6g.large** は ARM64 のタイプです。

2. このトピックの**ワーカーマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレート

は、クラスターに必要なネットワークオブジェクトおよびロードバランサーについて記述しています。

3. オプション: **m5** インスタンスタイプを **WorkerInstanceType** の値として指定した場合は、そのインスタンスタイプを CloudFormation テンプレートの **WorkerInstanceType.AllowedValues** パラメーターに追加します。

4. オプション: AWS Marketplace イメージを使用してデプロイする場合は、サブスクリプションから取得した AMI ID で **Worker0.type.properties.ImageID** パラメーターを更新します。

5. CloudFormation テンプレートを使用して、ワーカーノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

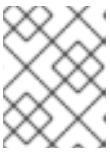
1 **<name>** は **cluster-worker-1** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-11eb-348f-sd9888c65b59
```



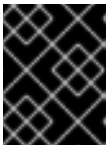
注記

CloudFormation テンプレートは、1 つのワーカーノードを表すスタックを作成します。

6. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

7. クラスターに作成するワーカーマシンが十分な数に達するまでワーカースタックの作成を継続します。同じテンプレートおよびパラメーターファイルを参照し、異なるスタック名を指定してワーカースタックをさらに作成することができます。



重要

2 つ以上のワーカーマシンを作成する必要があるため、この CloudFormation テンプレートを使用する 2 つ以上のスタックを作成する必要があります。

6.4.3.11.1. ワーカーマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例6.77 ワーカーマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the worker nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The worker security group ID to associate with worker nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with worker nodes.
    Type: String
  WorkerInstanceType:
    Default: m5.large
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
```

```
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

Outputs:
```

```
PrivateIP:
  Description: The compute node private IP address.
  Value: !GetAtt Worker0.PrivateIp
```

関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

6.4.3.12. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS でのブートストラップシーケンスの初期化

Amazon Web Services (AWS) ですべての必要なインフラストラクチャーを作成した後に、OpenShift Container Platform コントロールプレーンを初期化するブートストラップシーケンスを開始できます。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。
- コントロールプレーンマシンを作成している。
- ワーカーノードを作成している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、OpenShift Container Platform コントロールプレーンを初期化するブートストラッププロセスを開始します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level=info 2
```

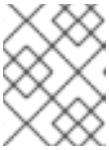
1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.mycluster.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 1s
```

コマンドが **FATAL** 警告を出さずに終了する場合、OpenShift Container Platform コントロールプレーンは初期化されています。



注記

コントロールプレーンの初期化後に、コンピュートノードを設定し、Operator の形式で追加のサービスをインストールします。

関連情報

- OpenShift Container Platform のインストールの進行中にインストール、ブートストラップ、コントロールプレーンのログを監視する方法の詳細は、インストールの進捗の監視 を参照してください。
- ブートストラッププロセスに関する問題のトラブルシューティングの詳細は、ブートストラップノードの診断データの収集 を参照してください。
- AWS EC2 コンソールを使用して、作成される実行中のインスタンスについての詳細を表示できます。

6.4.3.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.4.3.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

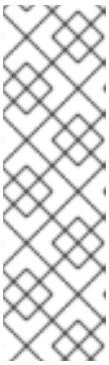
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
```

```
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.29.4
   master-1  Ready     master  73m  v1.29.4
   master-2  Ready     master  74m  v1.29.4
   worker-0  Ready     worker  11m  v1.29.4
   worker-1  Ready     worker  11m  v1.29.4
   ```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

6.4.3.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

6.4.3.15.1. イメージレジストリーストレージの設定

Amazon Web Services はデフォルトのストレージを提供します。つまり、Image Registry Operator はインストール後に利用可能になります。ただし、レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、レジストリーストレージを手動で設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

AWS のユーザーによってプロビジョニングされるインフラストラクチャーのレジストリーストレージを設定し、OpenShift Container Platform を非表示のリージョンにデプロイできます。詳細は、AWS user-provisioned infrastructure のレジストリー設定 を参照してください。

6.4.3.15.1.1. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS のレジストリーストレージの設定

インストール時に、Amazon S3 バケットを作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合、以下の手順により S3 バケットを作成し、ストレージを設定することができます。

前提条件

- user-provisioned infrastructure を使用した AWS 上にクラスターがある。
- Amazon S3 ストレージの場合、シークレットには以下のキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

手順

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、以下の手順を使用してください。

1. バケットライフサイクルポリシー を設定し、1 日以上経過している未完了のマルチパートアップロードを中止します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

警告

AWS でレジストリーイメージのセキュリティーを保護するには、S3 バケットに対して パブリックアクセスのブロック を実行します。

6.4.3.15.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

6.4.3.16. ブートストラップリソースの削除

クラスターの初期 Operator 設定の完了後に、Amazon Web Services (AWS) からブートストラップリソースを削除します。

前提条件

- クラスターの初期 Operator 設定が完了済みです。

手順

1. ブートストラップリソースを削除します。CloudFormation テンプレートを使用した場合は、そのスタックを削除 します。
   - AWS CLI を使用してスタックを削除します。

```
$ aws cloudformation delete-stack --stack-name <name> 1
```

1 **<name>** は、ブートストラップスタックの名前です。

   - AWS CloudFormation コンソール を使用してスタックを削除します。

6.4.3.17. Ingress DNS レコードの作成

DNS ゾーン設定を削除した場合には、Ingress ロードバランサーを参照する DNS レコードを手動で作成します。ワイルドカードレコードまたは特定のレコードのいずれかを作成できます。以下の手順では A レコードを使用しますが、CNAME やエイリアスなどの必要な他のレコードタイプを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用する OpenShift Container Platform クラスターを Amazon Web Services (AWS) にデプロイしています。
- OpenShift CLI (**oc**) がインストールされている。
- **jq** パッケージをインストールしている。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。

手順

1. 作成するルートを決定します。
   - ワイルドカードレコードを作成するには、***.apps.<cluster_name>.<domain_name>*** を使用します。ここで、**<cluster_name>** はクラスター名で、**<domain_name>** は OpenShift Container Platform クラスターの Route 53 ベースドメインです。
   - 特定のレコードを作成するには、以下のコマンドの出力にあるように、クラスターが使用する各ルートにレコードを作成する必要があります。

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
     {"\n"}{end}{end}' routes
     ```

     出力例

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
     prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
     ```

2. Ingress Operator ロードバランサーのステータスを取得し、使用する外部 IP アドレスの値をメモします。これは **EXTERNAL-IP** 列に表示されます。

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   出力例

   ```
   NAME              TYPE           CLUSTER-IP      EXTERNAL-IP                            PORT(S)
   AGE
   router-default    LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
   80:31499/TCP,443:30693/TCP    5m
   ```

3. ロードバランサーのホストゾーン ID を見つけます。

   ```
   $ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
   "<external_ip>").CanonicalHostedZoneNameID' 1
   ```

–

1. **<external_ip>** については、取得した Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。

出力例

```
Z3AADJGX6KTTL2
```

このコマンドの出力は、ロードバランサーのホストゾーン ID です。

4. クラスターのドメインのパブリックホストゾーン ID を取得します。

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ 1
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
        --output text
```

1 2 **<domain_name>** については、OpenShift Container Platform クラスターの Route 53 ベースドメインを指定します。

出力例

```
/hostedzone/Z3URY6TWQ91KVV
```

ドメインのパブリックホストゾーン ID がコマンド出力に表示されます。この例では、これは **Z3URY6TWQ91KVV** になります。

5. プライベートゾーンにエイリアスレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1. **<private_hosted_zone_id>** については、DNS および負荷分散の CloudFormation テンプレートの出力から値を指定します。

2. **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するド

3 **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。

4 **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

6. パブリックゾーンにレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1 **<public_hosted_zone_id>** については、ドメインのパブリックホストゾーンを指定します。

2 **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。

3 **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。

4 **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

6.4.3.18. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS インストールの実行

Amazon Web Service (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後に、デプロイメントを完了するまでモニターします。

前提条件

- OpenShift Container Platform クラスターのブートストラップノードを、ユーザーによってプロビジョニングされた AWS インフラストラクチャーで削除している。
- **oc** CLI をインストールしていること。

手順

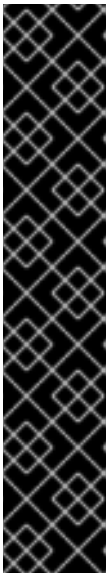
- インストールプログラムが含まれるディレクトリーから、クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 40m0s for the cluster at https://api.mycluster.example.com:6443 to
initialize...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 1s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

6.4.3.19. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

6.4.3.20. 関連情報

- AWS CloudFormation スタックについての詳細は、Working with stacks を参照してください。

6.4.3.21. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

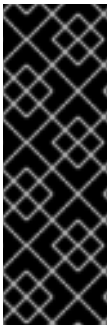
6.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、独自に提供するインフラストラクチャーとインストールリリースコンテンツの内部ミラーを使用して、Amazon Web Services (AWS) にクラスターをインストールできます。

重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが AWS API を使用するにはインターネットへのアクセスが必要になります。

このインフラストラクチャーを作成する 1 つの方法として、提供される CloudFormation テンプレートを使用できます。テンプレートを変更してインフラストラクチャーをカスタマイズしたり、それらに含まれる情報を使用し、所属する会社のポリシーに基づいて AWS オブジェクトを作成したりできます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の CloudFormation テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

6.4.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ミラーホストでミラーレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

  重要

  インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターをホストするために AWS アカウントを設定 している。

  重要

  AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ユーザーによってプロビジョニングされるインフラストラクチャーを準備している。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイトー覧も確認してください。

- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、長期間認証情報を手動で作成および維持 することができます。

6.4.4.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

6.4.4.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

6.4.4.3. AWS のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Amazon Web Services (AWS) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

6.4.4.3.1. オプション: 別個の /var パーティションの作成

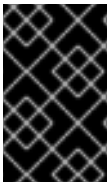
OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。



重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

6.4.4.3.2. インストール設定ファイルの作成

インストールプログラムがクラスターをデプロイするために必要なインストール設定ファイルを生成し、カスタマイズします。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャー用の OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- Red Hat が公開している付属の Red Hat Enterprise Linux CoreOS (RHCOS) AMI のある AWS リージョンにクラスターをデプロイしようとしている。カスタム AMI が必要な AWS リージョン (AWS GovCloud リージョンなど) にデプロイする場合は、 **install-config.yaml** ファイルを手動で作成する必要があります。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **aws** を選択します。

iii. AWS プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

注記

AWS アクセスキー ID およびシークレットアクセスキーは、インストールホストの現行ユーザーのホームディレクトリーの **~/.aws/credentials** に保存されます。エクスポートされたプロファイルの認証情報がファイルにない場合は、インストールプログラムにより認証情報の入力が求めるプロンプトが出されます。インストールプログラムに指定する認証情報は、ファイルに保存されます。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<local_registry> については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミ

ラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

c. イメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

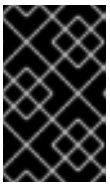
コマンドの出力の **imageContentSources** セクションを使用して、リポジトリー、またはネットワークが制限されたネットワークに取り込んだメディアからのコンテンツをミラーリングする際に使用した値をミラーリングします。

d. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

3. オプション: **install-config.yaml** ファイルをバックアップします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- AWS プロファイルおよび認証情報の設定についての詳細は、AWS ドキュメントの Configuration and credential file settings を参照してください。

6.4.4.3.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

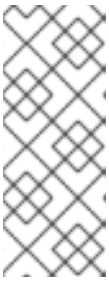
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。Amazon **EC2**、**Elastic Load Balancing**、および **S3** VPC エンドポイントを VPC に追加した場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの

trustedCA フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.4.4.3.4. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

重要

ユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールするときに **MachineAPI** 機能を無効にした場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除する必要があります。そうしないと、クラスターのインストールに失敗します。

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- 長期認証情報を手動で作成する

6.4.4.4. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Amazon Web Services (AWS) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な AWS リソースを見つけるためにも使用されます。提供される CloudFormation テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

6.4.4.5. AWS での VPC の作成

OpenShift Container Platform クラスターで使用する Virtual Private Cloud (VPC) を Amazon Web Services (AWS) で作成する必要があります。VPN およびルートテーブルを含む、各種要件を満たすように VPC をカスタマイズできます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、VPC を表す AWS リソースのスタックを作成できます。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1. VPC の CIDR ブロック。
2. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
3. VPC をデプロイするアベイラビリティーゾーンの数。
4. **1** から **3** の間の整数を指定します。
5. 各アベイラビリティーゾーン内の各サブネットのサイズ。
6. **5** から **13** の間の整数を指定します。 ここで、**5** は **/27** であり、 **13** は **/19** です。

2. このトピックの**VPC の CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

3. CloudFormation テンプレートを起動し、VPC を表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1. **<name>** は **cluster-vpc** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-vpc/dbedae40-2fd3-11eb-820e-12a48460849f
```

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

VpcId	VPC の ID。
PublicSubnetIds	新規パブリックサブネットの ID。
PrivateSubnetIds	新規プライベートサブネットの ID。

6.4.4.5.1. VPC の CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例6.78 VPC の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs
```

```
Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 =
/19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
```

```
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
```

```
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet3
    RouteTableId: !Ref PublicRouteTable
PrivateSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 0
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
NAT:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP
      - AllocationId
    SubnetId: !Ref PublicSubnet
EIP:
  Type: "AWS::EC2::EIP"
  Properties:
    Domain: vpc
Route:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT
PrivateSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 1
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable2:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
```

```
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable2
  NAT2:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz2
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP2
        - AllocationId
      SubnetId: !Ref PublicSubnet2
  EIP2:
    Type: "AWS::EC2::EIP"
    Condition: DoAz2
    Properties:
      Domain: vpc
  Route2:
    Type: "AWS::EC2::Route"
    Condition: DoAz2
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT2
  PrivateSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable3:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation3:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz3
    Properties:
      SubnetId: !Ref PrivateSubnet3
      RouteTableId: !Ref PrivateRouteTable3
  NAT3:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
```

```
    Condition: DoAz3
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
```

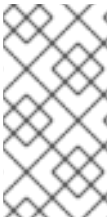
```
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
  PrivateRouteTableIds:
    Description: Private Route table IDs
    Value:
      !Join [
        ",",
        [
          !Join ["=", [
            !Select [0, "Fn::GetAZs": !Ref "AWS::Region"],
            !Ref PrivateRouteTable
          ]],
          !If [DoAz2,
              !Join ["=", [!Select [1, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable2]],
              !Ref "AWS::NoValue"
          ],
          !If [DoAz3,
              !Join ["=", [!Select [2, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable3]],
              !Ref "AWS::NoValue"
          ]
        ]
      ]
```

6.4.4.6. AWS でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用できるネットワークおよび負荷分散 (classic または network) を Amazon Web Services (AWS) で設定する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform クラスターに必要なネットワークおよび負荷分散コンポーネントを表します。テンプレートは、ホストゾーンおよびサブネットタグも作成します。

単一 Virtual Private Cloud 内でテンプレートを複数回実行することができます。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。

手順

1. クラスターの **install-config.yaml** ファイルに指定した Route 53 ベースドメインのホストゾーン ID を取得します。以下のコマンドを実行して、ホストゾーンの詳細を取得できます。

```
$ aws route53 list-hosted-zones-by-name --dns-name <route53_domain> 1
```

1 **<route53_domain>** について、クラスターの **install-config.yaml** ファイルを生成した時に作成した Route 53 ベースドメインを指定します。

出力例

```
mycluster.example.com. False 100
HOSTEDZONES 65F8F38E-2268-B835-E15C-AB55336FCBFA
/hostedzone/Z21IXYZABCZ2A4 mycluster.example.com. 10
```

この出力例では、ホストゾーン ID は **Z21IXYZ3-2Z2A4** です。

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "ClusterName", 1
    "ParameterValue": "mycluster" 2
  },
  {
    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
    "ParameterValue": "<random_string>" 6
  },
  {
    "ParameterKey": "HostedZoneName", 7
    "ParameterValue": "example.com" 8
  },
  {
    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
```

```
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]
```

1. ホスト名などに使用する短いクラスター名。
2. クラスターの **install-config.yaml** ファイルを生成した時に使用したクラスター名を指定します。
3. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
4. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
5. ターゲットの登録に使用する Route 53 パブリックトゾーン ID。
6. **Z21IXYZABCZ2A4** に類する形式の Route 53 パブリックゾーン ID を指定します。この値は AWS コンソールから取得できます。
7. ターゲットの登録に使用する Route 53 ゾーン。
8. クラスターの **install-config.yaml** ファイルを生成した時に使用した Route 53 ベースドメインを指定します。AWS コンソールに表示される末尾のピリド (.) は含めないでください。
9. VPC 用に作成したパブリックサブネット。
10. VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。
11. VPC 用に作成したプライベートサブネット。
12. VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。
13. クラスター用に作成した VPC。
14. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

3. このトピックの**ネットワークおよびロードバランサーの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。

重要

AWS government またはシークレットリージョンにクラスターをデプロイする場合は、CloudFormation テンプレートの **InternalApiServerRecord** を更新して、**CNAME** レコードを使用する必要があります。**ALIAS** タイプのレコードは、AWS 政府リージョンではサポートされません。

4. CloudFormation テンプレートを起動し、ネットワークおよび負荷分散コンポーネントを提供する AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>** は **cluster-dns** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。
4. 提供されるテンプレートは一部の **AWS::IAM::Role** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-dns/cd3e5de0-2fd4-11eb-5cf0-12be5c33a183
```

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

PrivateHostedZoneId	プライベート DNS のホストゾーン ID。
ExternalApiLoadBalancerName	外部 API ロードバランサーのフルネーム。
InternalApiLoadBalancerName	内部 API ロードバランサーのフルネーム。

ApiServerDnsName	API サーバーの完全ホスト名。
RegisterNlbIpTargetsLambda	これらのロードバランサーの登録/登録解除に役立つ Lambda ARN。
ExternalApiTargetGroupArn	外部 API ターゲットグループの ARN。
InternalApiTargetGroupArn	内部 API ターゲットグループの ARN。
InternalServiceTargetGroupArn	内部サービスターゲットグループの ARN。

6.4.4.6.1. ネットワークおよびロードバランサーの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例6.79 ネットワークおよびロードバランサーの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
```

```
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
        default: "Public Subnets"
      PrivateSubnets:
        default: "Private Subnets"
      HostedZoneName:
        default: "Public Hosted Zone Name"
      HostedZoneId:
        default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
```

```
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network

  IntDns:
    Type: "AWS::Route53::HostedZone"
    Properties:
      HostedZoneConfig:
        Comment: "Managed by CloudFormation"
      Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
      HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [!Ref InfrastructureName, "int"]]
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "owned"
      VPCs:
      - VPCId: !Ref VpcId
        VPCRegion: !Ref "AWS::Region"

  ExternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref HostedZoneId
      RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt ExtApiElb.DNSName

  InternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref IntDns
      RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
```

```
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
    - Name:
        !Join [
          ".",
          ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: ExternalApiTargetGroup
    LoadBalancerArn:
      Ref: ExtApiElb
    Port: 6443
    Protocol: TCP

ExternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

InternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalApiTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 6443
    Protocol: TCP
```

```
  InternalApiTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/readyz"
      HealthCheckPort: 6443
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 6443
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  InternalServiceInternalListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalServiceTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 22623
      Protocol: TCP

  InternalServiceTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/healthz"
      HealthCheckPort: 22623
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 22623
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  RegisterTargetLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
```

```
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalApiTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalServiceTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
```

```
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.8"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
              ]
            Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
```

```
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.8"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
    Value: !Ref ExternalApiTargetGroup
  InternalApiTargetGroupArn:
    Description: ARN of the internal API target group.
    Value: !Ref InternalApiTargetGroup
  InternalServiceTargetGroupArn:
    Description: ARN of the internal service target group.
    Value: !Ref InternalServiceTargetGroup
```

重要

クラスターを AWS government またはシークレットリージョンにデプロイする場合は、**InternalApiServerRecord** を更新し、**CNAME** レコードを使用する必要があります。**ALIAS** タイプのレコードは、AWS 政府リージョンではサポートされません。以下に例を示します。

```
Type: CNAME
TTL: 10
ResourceRecords:
- !GetAtt IntApiElb.DNSName
```

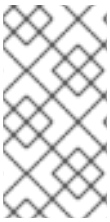
関連情報

- パブリックホストゾーンのリスト表示についての詳細は、AWS ドキュメントの Listing public hosted zones を参照してください。

6.4.4.7. AWS でのセキュリティーグループおよびロールの作成

OpenShift Container Platform クラスターで使用するセキュリティーグループおよびロールを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform クラスターに必要なセキュリティーグループおよびロールを表します。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
```

```
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. VPC の CIDR ブロック。
4. **x.x.x.x/16-24** の形式で定義した VPC に使用した CIDR ブロックパラメーターを指定します。
5. VPC 用に作成したプライベートサブネット。
6. VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。
7. クラスター用に作成した VPC。
8. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

2. このトピックの**セキュリティーオブジェクトの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なセキュリティーグループおよびロールについて記述しています。

3. CloudFormation テンプレートを起動し、セキュリティーグループおよびロールを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>** は **cluster-secs** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です

たはその名前です。

3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。
4. 提供されるテンプレートは一部の **AWS::IAM::Role** および **AWS::IAM::InstanceProfile** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-sec/03bd4210-2ed7-11eb-6d7a-13fc0b61e9db
```

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

MasterSecurityGroupId	マスターセキュリティーグループ ID
WorkerSecurityGroupId	ワーカーセキュリティーグループ ID
MasterInstanceProfile	マスター IAM インスタンスプロファイル
WorkerInstanceProfile	ワーカー IAM インスタンスプロファイル

6.4.4.7.1. セキュリティーオブジェクトの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なセキュリティーオブジェクトをデプロイすることができます。

例6.80 セキュリティーオブジェクトの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
```

```
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
```

```
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      ToPort: 6443
      FromPort: 6443
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      FromPort: 22623
      ToPort: 22623
      CidrIp: !Ref VpcCidr
    VpcId: !Ref VpcId

WorkerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Cluster Worker Security Group
    SecurityGroupIngress:
    - IpProtocol: icmp
      FromPort: 0
      ToPort: 0
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref VpcCidr
    VpcId: !Ref VpcId

MasterIngressEtcd:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: etcd
    FromPort: 2379
    ToPort: 2380
    IpProtocol: tcp

MasterIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressWorkerVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp
```

```
MasterIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressWorkerGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

MasterIngressIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

MasterIngressIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

MasterIngressWorkerIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
```

```
      IpProtocol: udp

  MasterIngressWorkerIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  MasterIngressWorkerIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  MasterIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressWorkerInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressWorkerInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
```

```
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

MasterIngressKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressWorkerKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressWorkerIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

MasterIngressWorkerIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
```

```
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressMasterVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressMasterGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
```

```
    IpProtocol: udp

WorkerIngressIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

WorkerIngressIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

WorkerIngressMasterIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

WorkerIngressMasterIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

WorkerIngressMasterIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

WorkerIngressInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
```

```
      IpProtocol: tcp

  WorkerIngressMasterInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressMasterInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes secure kubelet port
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal Kubernetes communication
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
```

```
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

  WorkerIngressMasterIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressMasterIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:AttachVolume"
```

```
            - "ec2:AuthorizeSecurityGroupIngress"
            - "ec2:CreateSecurityGroup"
            - "ec2:CreateTags"
            - "ec2:CreateVolume"
            - "ec2:DeleteSecurityGroup"
            - "ec2:DeleteVolume"
            - "ec2:Describe*"
            - "ec2:DetachVolume"
            - "ec2:ModifyInstanceAttribute"
            - "ec2:ModifyVolume"
            - "ec2:RevokeSecurityGroupIngress"
            - "elasticloadbalancing:AddTags"
            - "elasticloadbalancing:AttachLoadBalancerToSubnets"
            - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
            - "elasticloadbalancing:CreateListener"
            - "elasticloadbalancing:CreateLoadBalancer"
            - "elasticloadbalancing:CreateLoadBalancerPolicy"
            - "elasticloadbalancing:CreateLoadBalancerListeners"
            - "elasticloadbalancing:CreateTargetGroup"
            - "elasticloadbalancing:ConfigureHealthCheck"
            - "elasticloadbalancing:DeleteListener"
            - "elasticloadbalancing:DeleteLoadBalancer"
            - "elasticloadbalancing:DeleteLoadBalancerListeners"
            - "elasticloadbalancing:DeleteTargetGroup"
            - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
            - "elasticloadbalancing:DeregisterTargets"
            - "elasticloadbalancing:Describe*"
            - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
            - "elasticloadbalancing:ModifyListener"
            - "elasticloadbalancing:ModifyLoadBalancerAttributes"
            - "elasticloadbalancing:ModifyTargetGroup"
            - "elasticloadbalancing:ModifyTargetGroupAttributes"
            - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
            - "elasticloadbalancing:RegisterTargets"
            - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
            - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
            - "kms:DescribeKey"
          Resource: "*"

    MasterInstanceProfile:
      Type: "AWS::IAM::InstanceProfile"
      Properties:
        Roles:
        - Ref: "MasterIamRole"

    WorkerIamRole:
      Type: AWS::IAM::Role
      Properties:
        AssumeRolePolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Principal:
              Service:
              - "ec2.amazonaws.com"
            Action:
```

```
          - "sts:AssumeRole"
        Policies:
        - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
            - Effect: "Allow"
              Action:
              - "ec2:DescribeInstances"
              - "ec2:DescribeRegions"
              Resource: "*"

    WorkerInstanceProfile:
      Type: "AWS::IAM::InstanceProfile"
      Properties:
        Roles:
        - Ref: "WorkerIamRole"

  Outputs:
    MasterSecurityGroupId:
      Description: Master Security Group ID
      Value: !GetAtt MasterSecurityGroup.GroupId

    WorkerSecurityGroupId:
      Description: Worker Security Group ID
      Value: !GetAtt WorkerSecurityGroup.GroupId

    MasterInstanceProfile:
      Description: Master IAM Instance Profile
      Value: !Ref MasterInstanceProfile

    WorkerInstanceProfile:
      Description: Worker IAM Instance Profile
      Value: !Ref WorkerInstanceProfile
```

6.4.4.8. ストリームメタデータを使用した RHCOS AMI へのアクセス

OpenShift Container Platform では、**ストリームメタデータ** は、JSON 形式で RHCOS に関する標準化されたメタデータを 提供し、メタデータをクラスターに挿入します。ストリームメタデータは、複数のアーキテクチャーをサポートする安定した形式で、自動化を維持するための自己文書化が意図されています。

openshift-install の **coreos print-stream-json** サブコマンドを使用して、ストリームメタデータ形式のブートイメージに関する情報にアクセスできます。このコマンドは、スクリプト可能でマシン読み取り可能な形式でストリームメタデータを出力する方法を提供します。

ユーザーによってプロビジョニングされるインストールの場合、**openshift-install** バイナリーには、AWS AMI などの OpenShift Container Platform での使用がテストされている RHCOS ブートイメージのバージョンへの参照が含まれます。

手順

ストリームメタデータを解析するには、以下のいずれかの方法を使用します。

- Go プログラムから、https://github.com/coreos/stream-metadata-go の公式の **stream-metadata-go** ライブラリーを使用します。ライブラリーでサンプルコードを確認することもできます。
- Python や Ruby などの別のプログラミング言語から、お好みのプログラミング言語の JSON ライブラリーを使用します。
- **jq** などの JSON データを処理するコマンドラインユーティリティーから、以下のコマンドを実行します。
  - **us-west-1** などの AWS リージョンの現在の **x86_64** または **aarch64** AMI を出力します。

    x86_64 の場合

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.x86_64.images.aws.regions["us-west-1"].image'
    ```

    出力例

    ```
    ami-0d3e625f84626bbda
    ```

    aarch64 の場合

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.aarch64.images.aws.regions["us-west-1"].image'
    ```

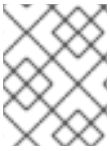
    出力例

    ```
    ami-0af1d3b7fa5be2131
    ```

    このコマンドの出力は、指定されたアーキテクチャーと **us-west-1** リージョンの AWS AMI ID です。AMI はクラスターと同じリージョンに属する必要があります。

6.4.4.9. AWS インフラストラクチャーの RHCOS AMI

Red Hat は、OpenShift Container Platform ノードに手動で指定できる、さまざまな AWS リージョンおよびインスタンスアーキテクチャーに有効な Red Hat Enterprise Linux CoreOS(RHCOS) AMI を提供します。



注記

また、独自の AMI をインポートすることで、RHCOS AMI がパブリッシュされていないリージョンにインストールすることもできます。

表6.25 x86_64 RHCOS AMIs

AWS ゾーン	AWS AMI
af-south-1	**ami-018de6b1be470b7e6**
ap-east-1	**ami-092f09a4c8aacf56a**

AWS ゾーン	AWS AMI
ap-northeast-1	ami-001c9fc85b77505f4
ap-northeast-2	ami-0a5d7f1966d88fba3
ap-northeast-3	ami-085a2726ceb4f5eeb
ap-south-1	ami-03f2c19071fb6eab3
ap-south-2	ami-0c82522890979d94b
ap-southeast-1	ami-06e5b9d16ead6c55c
ap-southeast-2	ami-0ccd429129fbf6bc0
ap-southeast-3	ami-096f9b4d41116d95c
ap-southeast-4	ami-0b511ab55f9f7d052
ca-central-1	ami-0e357287c651be1f2
ca-west-1	ami-0b1692e5776740901
eu-central-1	ami-08b13fb388ba5610d
eu-central-2	ami-04777298b55045ea9
eu-north-1	ami-05421993a4ee567b9
eu-south-1	ami-00959341869d34746
eu-south-2	ami-0a745b610522a87cc
eu-west-1	ami-0e48f85ec57bd7baa
eu-west-2	ami-0c015b1387acddc5e
eu-west-3	ami-01e466af77a95b93d
il-central-1	ami-0a1b706793b54d087
me-central-1	ami-09d58e8b2099ae5bc
me-south-1	ami-0f9c259bc4346599c

AWS ゾーン	AWS AMI
sa-east-1	**ami-06277517d966881a3**
us-east-1	**ami-05483066c3caaccf5**
us-east-2	**ami-0c704ce516493a479**
us-gov-east-1	**ami-0695b2cbdd1ec4d48**
us-gov-west-1	**ami-004404a0eaa427b39**
us-west-1	**ami-0aaa56637063be772**
us-west-2	**ami-0870c7a3d5ef4d2b3**

表6.26 aarch64 RHCOS AMI

AWS ゾーン	AWS AMI
af-south-1	**ami-0d96d447d3df14f4e**
ap-east-1	**ami-010236402dfba1717**
ap-northeast-1	**ami-08feeb6d9068bb12e**
ap-northeast-2	**ami-0772082c119147205**
ap-northeast-3	**ami-05bcbb13493d0516f**
ap-south-1	**ami-0034a539e8adcb817**
ap-south-2	**ami-0fc56b7c53c38ff70**
ap-southeast-1	**ami-0b50a0e92a58f3ad8**
ap-southeast-2	**ami-0697a9174ae206182**
ap-southeast-3	**ami-05ae309319a7ad504**
ap-southeast-4	**ami-00500414843baa657**
ca-central-1	**ami-05e76bf99d3b0d4c8**
ca-west-1	**ami-099fc953c221ec590**

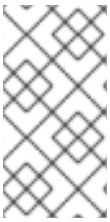
AWS ゾーン	AWS AMI
eu-central-1	**ami-0d2e2698884020b71**
eu-central-2	**ami-0a96ba21ddee01719**
eu-north-1	**ami-0ab8a1070d0b1d9a5**
eu-south-1	**ami-0d0465299a8b196c1**
eu-south-2	**ami-07d5aa3c6d468c738**
eu-west-1	**ami-08e7d209f26c09c80**
eu-west-2	**ami-0c8336d4e2c1f13cb**
eu-west-3	**ami-085d804b98acf0648**
il-central-1	**ami-02ce030e36aeb7643**
me-central-1	**ami-0dea3ac466040b77f**
me-south-1	**ami-02ef2a46887b9786d**
sa-east-1	**ami-0d19817d31b2399f9**
us-east-1	**ami-04859c888566a2c2f**
us-east-2	**ami-02f7e4a5dc66361bb**
us-gov-east-1	**ami-067ef782980ea96ad**
us-gov-west-1	**ami-0ce67540c1bb2319d**
us-west-1	**ami-0a09c5d2f287718a3**
us-west-2	**ami-06b94e64e595f6cee**

6.4.4.10. AWS でのブートストラップノードの作成

OpenShift Container Platform クラスターの初期化で使用するブートストラップノードを Amazon Web Services (AWS) で作成する必要があります。これは、以下の方法で行います。

- **bootstrap.ign** Ignition 設定ファイルをクラスターに送るための場所を指定。このファイルはインストールディレクトリーに置かれます。提供される CloudFormation テンプレートでは、クラスターの Ignition 設定ファイルは S3 バケットから送られることを前提としています。このファイルを別の場所から送ることを選択する場合は、テンプレートを変更する必要があります。

- 提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform インストールに必要なブートストラップノードを表します。



注記

提供される CloudFormation テンプレートを使用してブートストラップノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。

手順

1. 以下のコマンドを実行してバケットを作成します。

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra** はバケット名です。**install-config.yaml** ファイルを作成する際に、**<cluster-name>** をクラスターに指定された名前に置き換えます。

以下の場合は、**s3://** スキーマではなく、S3 バケットに事前に署名された URL を使用する必要があります。

- AWS SDK とは異なるエンドポイントを持つリージョンへのデプロイ。
- プロキシーをデプロイする。
- カスタムエンドポイントを指定します。

2. 以下のコマンドを実行して **bootstrap.ign** Ignition 設定ファイルをバケットにアップロードします。

```
$ aws s3 cp <installation_directory>/bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign 1
```

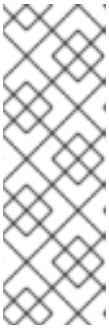
1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

3. 以下のコマンドを実行して、ファイルがアップロードされていることを確認します。

```
$ aws s3 ls s3://<cluster-name>-infra/
```

出力例

```
2019-04-03 16:15:16     314878 bootstrap.ign
```



注記

ブートストラップ Ignition 設定ファイルには、X.509 キーのようなシークレットが含まれません。以下の手順では、S3 バケットの基本的なセキュリティーを提供します。追加のセキュリティーを提供するには、OpenShift IAM ユーザーなどの特定のユーザーのみがバケットに含まれるオブジェクトにアクセスできるように S3 バケットポリシーを有効にできます。S3 を完全に回避し、ブートストラップマシンが到達できるアドレスからブートストラップ Ignition 設定ファイルを送ることができます。

4. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 9
    "ParameterValue": "sg-<random_string>" 10
  },
  {
    "ParameterKey": "VpcId", 11
    "ParameterValue": "vpc-<random_string>" 12
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", 13
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
  },
  {
    "ParameterKey": "AutoRegisterELB", 15
    "ParameterValue": "yes" 16
  },
  {
```

```
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
    "ParameterValue": "arn:aws:lambda:<aws_region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 19
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. 選択したアーキテクチャーに基づいてブートストラップノードに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. 有効な **AWS::EC2::Image::Id** 値を指定します。
5. ブートストラップノードへの SSH アクセスを許可する CIDR ブロック。
6. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
7. ブートストラップを起動するために VPC に関連付けられるパブリックサブネット。
8. VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。
9. マスターセキュリティーグループ ID (一時ルールの登録用)。
10. セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。
11. 作成されたリソースが属する VPC。
12. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。
13. ブートストラップの Ignition 設定ファイルをフェッチする場所。
14. **s3://<bucket_name>/bootstrap.ign** の形式で S3 バケットおよびファイル名を指定します。
15. ネットワークロードバランサー (NLB) を登録するかどうか。

16 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

17 NLB IP ターゲット登録 lambda グループの ARN。

18 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

19 外部 API ロードバランサーのターゲットグループの ARN。

20 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

21 内部 API ロードバランサーのターゲットグループの ARN。

22 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

23 内部サービスバランサーのターゲットグループの ARN。

24 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

5. このトピックの**ブートストラップマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。

6. オプション： プロキシーを使用してクラスターをデプロイする場合は、テンプレートの ignition を更新して **ignition.config.proxy** フィールドを追加する必要があります。さらに、Amazon EC2、Elastic Load Balancing、および S3 VPC エンドポイントを VPC に追加している場合は、これらのエンドポイントを **noProxy** フィールドに追加する必要があります。

7. CloudFormation テンプレートを起動し、ブートストラップノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>** は **cluster-bootstrap** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4 提供されるテンプレートは一部の **AWS::IAM::Role** および **AWS::IAM::InstanceProfile** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-bootstrap/12944486-2add-11eb-9dee-12dace8e3a83
```

8. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

Bootstrap InstanceId	ブートストラップインスタンス ID。
Bootstrap PublicIp	ブートストラップノードのパブリック IP アドレス。
Bootstrap PrivateIp	ブートストラップノードのプライベート IP アドレス。

6.4.4.10.1. ブートストラップマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイできます。

例6.81 ブートストラップマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
```

```
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String
  BootstrapInstanceType:
    Description: Instance type for the bootstrap EC2 instance
    Default: "i3.large"
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
```

```
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      AllowedBootstrapSshCidr:
        default: "Allowed SSH Source"
      PublicSubnet:
        default: "Public Subnet"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Bootstrap Ignition Source"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  BootstrapIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
```

```
          Action: "ec2:Describe*"
          Resource: "*"
        - Effect: "Allow"
          Action: "ec2:AttachVolume"
          Resource: "*"
        - Effect: "Allow"
          Action: "ec2:DetachVolume"
          Resource: "*"
        - Effect: "Allow"
          Action: "s3:GetObject"
          Resource: "*"

BootstrapInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Path: "/"
    Roles:
    - Ref: "BootstrapIamRole"

BootstrapSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Cluster Bootstrap Security Group
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref AllowedBootstrapSshCidr
    - IpProtocol: tcp
      ToPort: 19531
      FromPort: 19531
      CidrIp: 0.0.0.0/0
    VpcId: !Ref VpcId

BootstrapInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    IamInstanceProfile: !Ref BootstrapInstanceProfile
    InstanceType: !Ref BootstrapInstanceType
    NetworkInterfaces:
    - AssociatePublicIpAddress: "true"
      DeviceIndex: "0"
      GroupSet:
      - !Ref "BootstrapSecurityGroup"
      - !Ref "MasterSecurityGroupId"
      SubnetId: !Ref "PublicSubnet"
    UserData:
      Fn::Base64: !Sub
      - '{"ignition":{"config":{"replace":{"source":"${S3Loc}"}},"version":"3.1.0"}}'
      - {
        S3Loc: !Ref BootstrapIgnitionLocation
      }

RegisterBootstrapApiTarget:
  Condition: DoRegistration
```

```
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref ExternalApiTargetGroupArn
        TargetIp: !GetAtt BootstrapInstance.PrivateIp

    RegisterBootstrapInternalApiTarget:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref InternalApiTargetGroupArn
        TargetIp: !GetAtt BootstrapInstance.PrivateIp

    RegisterBootstrapInternalServiceTarget:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref InternalServiceTargetGroupArn
        TargetIp: !GetAtt BootstrapInstance.PrivateIp

  Outputs:
    BootstrapInstanceId:
      Description: Bootstrap Instance ID.
      Value: !Ref BootstrapInstance

    BootstrapPublicIp:
      Description: The bootstrap node public IP address.
      Value: !GetAtt BootstrapInstance.PublicIp

    BootstrapPrivateIp:
      Description: The bootstrap node private IP address.
      Value: !GetAtt BootstrapInstance.PrivateIp
```

関連情報

- AWS ゾーンの Red Hat Enterprise Linux CoreOS (RHCOS) AMI についての詳細は、 AWS インフラストラクチャーの RHCOS AMI を参照してください。

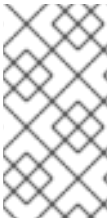
6.4.4.11. AWS でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、コントロールプレーンノードを表す AWS リソースのスタックを作成できます。

重要

CloudFormation テンプレートは、3 つのコントロールプレーンノードを表すスタックを作成します。



注記

提供される CloudFormation テンプレートを使用してコントロールプレーンノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
```

```
  "ParameterKey": "Master1Subnet", (13)
  "ParameterValue": "subnet-<random_string>" (14)
 },
 {
  "ParameterKey": "Master2Subnet", (15)
  "ParameterValue": "subnet-<random_string>" (16)
 },
 {
  "ParameterKey": "MasterSecurityGroupId", (17)
  "ParameterValue": "sg-<random_string>" (18)
 },
 {
  "ParameterKey": "IgnitionLocation", (19)
  "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
(20)
 },
 {
  "ParameterKey": "CertificateAuthorities", (21)
  "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" (22)
 },
 {
  "ParameterKey": "MasterInstanceProfileName", (23)
  "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" (24)
 },
 {
  "ParameterKey": "MasterInstanceType", (25)
  "ParameterValue": "" (26)
 },
 {
  "ParameterKey": "AutoRegisterELB", (27)
  "ParameterValue": "yes" (28)
 },
 {
  "ParameterKey": "RegisterNlbIpTargetsLambdaArn", (29)
  "ParameterValue": "arn:aws:lambda:<aws_region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" (30)
 },
 {
  "ParameterKey": "ExternalApiTargetGroupArn", (31)
  "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" (32)
 },
 {
  "ParameterKey": "InternalApiTargetGroupArn", (33)
  "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" (34)
 },
 {
  "ParameterKey": "InternalServiceTargetGroupArn", (35)
  "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" (36)
 }
]
```

```
-
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. 選択したアーキテクチャーに基づいてコントロールプレーンマシンに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. DNS etcd 登録を実行するかどうか。
6. **yes** または **no** を指定します。**yes** を指定する場合、ホストゾーンの情報を指定する必要があります。
7. etcd ターゲットの登録に使用する Route 53 プライベートゾーン ID。
8. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateHostedZoneId** 値を指定します。
9. ターゲットの登録に使用する Route 53 ゾーン。
10. **<cluster_name>.<domain_name>** を指定します。ここで、**<domain_name>** はクラスターの **install-config.yaml** ファイルの生成時に使用した Route 53 ベースドメインです。AWS コンソールに表示される末尾のピリド (.) は含めないでください。
11. 13. 15. コントロールプレーンマシンの起動に使用するサブネット (プライベートが望ましい)。
12. 14. 16. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。
17. コントロールプレーンノードに関連付けるマスターセキュリティーグループ ID。
18. セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。
19. コントロールプレーンの Ignition 設定ファイルをフェッチする場所。
20. 生成される Ignition 設定ファイルの場所を指定します (**https://api-int.<cluster_name>.<domain_name>:22623/config/master**)。
21. 使用する base64 でエンコードされた認証局の文字列。
22. インストールディレクトリーにある **master.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。
23. コントロールプレーンノードに関連付ける IAM プロファイル。
24. セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **MasterInstanceProfile** パラメーターの値を指定します。
25. 選択したアーキテクチャーに基づいてコントロールプレーンマシンに使用する AWS インスタンスのタイプ。
26.

インスタンスタイプの値は、コントロールプレーンマシンの最小リソース要件に対応します。たとえば、**m6i.xlarge** は AMD64 のタイプであり、**m6g.xlarge** は、ARM64 のタイプ

27 ネットワークロードバランサー (NLB) を登録するかどうか。

28 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

29 NLB IP ターゲット登録 lambda グループの ARN。

30 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

31 外部 API ロードバランサーのターゲットグループの ARN。

32 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

33 内部 API ロードバランサーのターゲットグループの ARN。

34 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

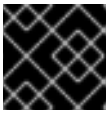
35 内部サービスバランサーのターゲットグループの ARN。

36 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

2. このトピックの**コントロールプレーンマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。

3. **m5** インスタンスタイプを **MasterInstanceType** の値として指定している場合、そのインスタンスタイプを CloudFormation テンプレートの **MasterInstanceType.AllowedValues** パラメーターに追加します。

4. CloudFormation テンプレートを起動し、コントロールプレーンノードを表す AWS リソースのスタックを作成します。



重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-control-plane** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-control-plane/21c7e2b0-2ee2-11eb-c6f6-0aa34627df4b
```

注記

CloudFormation テンプレートは、3 つのコントロールプレーンノードを表すスタックを作成します。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

6.4.4.11.1. コントロールプレーンマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例6.82 コントロールプレーンマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: ""
    Description: unused
    Type: String
  PrivateHostedZoneId:
    Default: ""
    Description: unused
    Type: String
  PrivateHostedZoneName:
    Default: ""
    Description: unused
```

```
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m5.xlarge
    Type: String

  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
```

```
      default: "Cluster Information"
    Parameters:
    - InfrastructureName
  - Label:
      default: "Host Information"
    Parameters:
    - MasterInstanceType
    - RhcosAmi
    - IgnitionLocation
    - CertificateAuthorities
    - MasterSecurityGroupId
    - MasterInstanceProfileName
  - Label:
      default: "Network Configuration"
    Parameters:
    - VpcId
    - AllowedBootstrapSshCidr
    - Master0Subnet
    - Master1Subnet
    - Master2Subnet
  - Label:
      default: "Load Balancer Automation"
    Parameters:
    - AutoRegisterELB
    - RegisterNlbIpTargetsLambdaArn
    - ExternalApiTargetGroupArn
    - InternalApiTargetGroupArn
    - InternalServiceTargetGroupArn
  ParameterLabels:
    InfrastructureName:
      default: "Infrastructure Name"
    VpcId:
      default: "VPC ID"
    Master0Subnet:
      default: "Master-0 Subnet"
    Master1Subnet:
      default: "Master-1 Subnet"
    Master2Subnet:
      default: "Master-2 Subnet"
    MasterInstanceType:
      default: "Master Instance Type"
    MasterInstanceProfileName:
      default: "Master Instance Profile Name"
    RhcosAmi:
      default: "Red Hat Enterprise Linux CoreOS AMI ID"
    BootstrapIgnitionLocation:
      default: "Master Ignition Source"
    CertificateAuthorities:
      default: "Ignition CA String"
    MasterSecurityGroupId:
      default: "Master Security Group ID"
    AutoRegisterELB:
      default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
```

```
Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
```

```
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
```

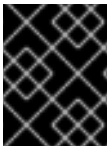
```
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
```

```
      ",",
      [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
    ]
```

6.4.4.12. AWS でのワーカーノードの作成

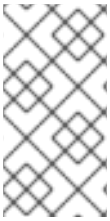
クラスターで使用するワーカーノードを Amazon Web Services (AWS) で作成できます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、ワーカーノードを表す AWS リソースのスタックを作成できます。



重要

CloudFormation テンプレートは、1 つのワーカーノードを表すスタックを作成します。それぞれのワーカーノードにスタックを作成する必要があります。



注記

提供される CloudFormation テンプレートを使用してワーカーノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。
- コントロールプレーンマシンを作成している。

手順

1. CloudFormation テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
```

```
  "ParameterKey": "RhcosAmi", 3
  "ParameterValue": "ami-<random_string>" 4
 },
 {
  "ParameterKey": "Subnet", 5
  "ParameterValue": "subnet-<random_string>" 6
 },
 {
  "ParameterKey": "WorkerSecurityGroupId", 7
  "ParameterValue": "sg-<random_string>" 8
 },
 {
  "ParameterKey": "IgnitionLocation", 9
  "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
 },
 {
  "ParameterKey": "CertificateAuthorities", 11
  "ParameterValue": "" 12
 },
 {
  "ParameterKey": "WorkerInstanceProfileName", 13
  "ParameterValue": "" 14
 },
 {
  "ParameterKey": "WorkerInstanceType", 15
  "ParameterValue": "" 16
 }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. 選択したアーキテクチャーに基づいてワーカーノードに使用する現在の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. ワーカーノードを起動するためのサブネット (プライベートであることが望ましい)。
6. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。
7. ワーカーノードに関連付けるワーカーセキュリティーグループ ID。
8. セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WorkerSecurityGroupId** 値を指定します。
9. ブートストラップ Ignition 設定ファイルを取得する場所。
10. 生成される Ignition 設定の場所を指定します。 **https://api-int.<cluster_name>.<domain_name>:22623/config/worker**

11 使用する base64 でエンコードされた認証局の文字列。

12 インストールディレクトリーにある **worker.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

13 ワーカーロールに関連付ける IAM プロファイル。

14 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WokerInstanceProfile** パラメーターの値を指定します。

15 選択したアーキテクチャーに基づいてコンピュートマシンに使用する AWS インスタンスのタイプ。

16 インスタンスタイプの値は、コンピュートマシンの最小リソース要件に対応します。たとえば、**m6i.large** は AMD64 のタイプであり、**m6g.large** は ARM64 のタイプです。

2. このトピックの**ワーカーマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークオブジェクトおよびロードバランサーについて記述しています。

3. オプション: **m5** インスタンスタイプを **WorkerInstanceType** の値として指定した場合は、そのインスタンスタイプを CloudFormation テンプレートの **WorkerInstanceType.AllowedValues** パラメーターに追加します。

4. オプション: AWS Marketplace イメージを使用してデプロイする場合は、サブスクリプションから取得した AMI ID で **Worker0.type.properties.ImageID** パラメーターを更新します。

5. CloudFormation テンプレートを使用して、ワーカーノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-worker-1** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-11eb-348f-sd9888c65b59
```

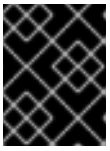
注記

CloudFormation テンプレートは、1 つのワーカーノードを表すスタックを作成します。

6. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

7. クラスターに作成するワーカーマシンが十分な数に達するまでワーカースタックの作成を継続します。同じテンプレートおよびパラメーターファイルを参照し、異なるスタック名を指定してワーカースタックをさらに作成することができます。



重要

2 つ以上のワーカーマシンを作成する必要があるため、この CloudFormation テンプレートを使用する 2 つ以上のスタックを作成する必要があります。

6.4.4.12.1. ワーカーマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例6.83 ワーカーマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the worker nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The worker security group ID to associate with worker nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
```

```
  WorkerInstanceProfileName:
    Description: IAM profile to associate with worker nodes.
    Type: String
  WorkerInstanceType:
    Default: m5.large
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
```

```
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

6.4.4.13. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS でのブートストラップシーケンスの初期化

Amazon Web Services (AWS) ですべての必要なインフラストラクチャーを作成した後に、OpenShift Container Platform コントロールプレーンを初期化するブートストラップシーケンスを開始できます。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。
- コントロールプレーンマシンを作成している。
- ワーカーノードを作成している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、OpenShift Container Platform コントロールプレーンを初期化するブートストラッププロセスを開始します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level=info 2
```

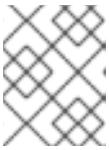
1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.mycluster.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 1s
```

コマンドが **FATAL** 警告を出さずに終了する場合、OpenShift Container Platform コントロールプレーンは初期化されています。



注記

コントロールプレーンの初期化後に、コンピュートノードを設定し、Operator の形式で追加のサービスをインストールします。

関連情報

- OpenShift Container Platform のインストールの進行中にインストール、ブートストラップ、コントロールプレーンのログを監視する方法の詳細は、インストールの進捗の監視 を参照してください。
- ブートストラッププロセスに関する問題のトラブルシューティングの詳細は、ブートストラップノードの診断データの収集 を参照してください。

6.4.4.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

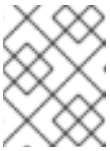
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

6.4.4.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                            4.16.0    True        False         False      40m
cluster-autoscaler                          4.16.0    True        False         False      37m
config-operator                             4.16.0    True        False         False      38m
```

```
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

6.4.4.15.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

6.4.4.15.2. イメージレジストリーストレージの設定

Amazon Web Services はデフォルトのストレージを提供します。つまり、Image Registry Operator はインストール後に利用可能になります。ただし、レジストリー Operator が S3 バケットを作成でき

ず、ストレージを自動的に設定する場合は、レジストリーストレージを手動で設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

6.4.4.15.2.1. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS のレジストリーストレージの設定

インストール時に、Amazon S3 バケットを作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合、以下の手順により S3 バケットを作成し、ストレージを設定することができます。

前提条件

- user-provisioned infrastructure を使用した AWS 上にクラスターがある。
- Amazon S3 ストレージの場合、シークレットには以下のキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

手順

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、以下の手順を使用してください。

1. バケットライフサイクルポリシー を設定し、1 日以上経過している未完了のマルチパートアップロードを中止します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

警告

AWS でレジストリーイメージのセキュリティーを保護するには、S3 バケットに対して パブリックアクセスのブロック を実行します。

6.4.4.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

6.4.4.16. ブートストラップリソースの削除

クラスターの初期 Operator 設定の完了後に、Amazon Web Services (AWS) からブートストラップリソースを削除します。

前提条件

- クラスターの初期 Operator 設定が完了済みです。

手順

1. ブートストラップリソースを削除します。CloudFormation テンプレートを使用した場合は、そのスタックを削除 します。
   - AWS CLI を使用してスタックを削除します。
     -

```
$ aws cloudformation delete-stack --stack-name <name> 1
```

1 **<name>** は、ブートストラップスタックの名前です。

- AWS CloudFormation コンソール を使用してスタックを削除します。

6.4.4.17. Ingress DNS レコードの作成

DNS ゾーン設定を削除した場合には、Ingress ロードバランサーを参照する DNS レコードを手動で作成します。ワイルドカードレコードまたは特定のレコードのいずれかを作成できます。以下の手順では A レコードを使用しますが、CNAME やエイリアスなどの必要な他のレコードタイプを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用する OpenShift Container Platform クラスターを Amazon Web Services (AWS) にデプロイしています。
- OpenShift CLI (**oc**) がインストールされている。
- **jq** パッケージをインストールしている。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。

手順

1. 作成するルートを決定します。
   - ワイルドカードレコードを作成するには、***.apps.<cluster_name>.<domain_name>** を使用します。ここで、**<cluster_name>** はクラスター名で、**<domain_name>** は OpenShift Container Platform クラスターの Route 53 ベースドメインです。
   - 特定のレコードを作成するには、以下のコマンドの出力にあるように、クラスターが使用する各ルートにレコードを作成する必要があります。

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
     {"\n"}{end}{end}' routes
     ```

     出力例

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
     prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
     ```

2. Ingress Operator ロードバランサーのステータスを取得し、使用する外部 IP アドレスの値をメモします。これは **EXTERNAL-IP** 列に表示されます。

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   出力例

```
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP                          PORT(S)
AGE
router-default    LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
80:31499/TCP,443:30693/TCP   5m
```

3. ロードバランサーのホストゾーン ID を見つけます。

```
$ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
"<external_ip>").CanonicalHostedZoneNameID' 1
```

1 **<external_ip>** については、取得した Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。

出力例

```
Z3AADJGX6KTTL2
```

このコマンドの出力は、ロードバランサーのホストゾーン ID です。

4. クラスターのドメインのパブリックホストゾーン ID を取得します。

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ 1
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
        --output text
```

1 2 **<domain_name>** については、OpenShift Container Platform クラスターの Route 53 ベースドメインを指定します。

出力例

```
/hostedzone/Z3URY6TWQ91KVV
```

ドメインのパブリックホストゾーン ID がコマンド出力に表示されます。この例では、これは **Z3URY6TWQ91KVV** になります。

5. プライベートゾーンにエイリアスレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
```

```
>       }
>     }
>   }
> ]
>}'
```

1. **<private_hosted_zone_id>** については、DNS および負荷分散の CloudFormation テンプレートの出力から値を指定します。
2. **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。
3. **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。
4. **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

6. パブリックゾーンにレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ (1)
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", (2)
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", (3)
>           "DNSName": "<external_ip>.", (4)
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
>}'
```

1. **<public_hosted_zone_id>** については、ドメインのパブリックホストゾーンを指定します。
2. **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。
3. **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。
4. **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

6.4.4.18. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS インストールの実行

Amazon Web Service (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後に、デプロイメントを完了するまでモニターします。

前提条件

- OpenShift Container Platform クラスターのブートストラップノードを、ユーザーによってプロビジョニングされた AWS インフラストラクチャーで削除している。
- **oc** CLI をインストールしていること。

手順

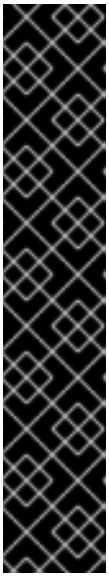
1. インストールプログラムが含まれるディレクトリーから、クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 40m0s for the cluster at https://api.mycluster.example.com:6443 to
initialize...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 1s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Cluster registration ページでクラスターを登録します。

6.4.4.19. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.4.4.20. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

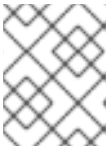
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

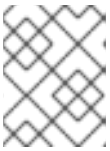


注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

6.4.4.21. 関連情報

- AWS CloudFormation スタックについての詳細は、Working with stacks を参照してください。

6.4.4.22. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。

- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

6.5. AWS に 3 ノードクラスターをインストールする

OpenShift Container Platform バージョン 4.16 では、Amazon Web Services (AWS) に 3 ノードのクラスターをインストールできます。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコントロールプレーンマシンで設定されます。このタイプのクラスターは、クラスター管理者および開発者がテスト、開発、および実稼働に使用するためのより小さくリソース効率の高いクラスターを提供します。

インストーラーによってプロビジョニングされたインフラストラクチャーまたはユーザーによってプロビジョニングされたインフラストラクチャーのいずれかを使用して、3 ノードクラスターをインストールできます。

注記

AWS Marketplace イメージを使用した 3 ノードクラスターのデプロイはサポートされていません。

6.5.1. 3 ノードクラスターの設定

クラスターをデプロイする前に、**install-config.yaml** ファイルでワーカーノードの数を **0** に設定して、3 ノードクラスターを設定します。ワーカーノードの数を **0** に設定すると、コントロールプレーンマシンがスケジュール可能になります。これにより、アプリケーションワークロードをコントロールプレーンノードから実行するようにスケジュールできます。

注記

アプリケーションワークロードはコントロールプレーンノードから実行され、コントロールプレーンノードはコンピュートノードと見なされるため、追加のサブスクリプションが必要です。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

1. 次の **compute** スタンザに示すように、**install-config.yaml** ファイルでコンピューティングレプリカの数を **0** に設定します。

   3 ノードクラスターの install-config.yaml ファイルの例

   ```
   apiVersion: v1
   baseDomain: example.com
   compute:
   - name: worker
     platform: {}
     replicas: 0
   # ...
   ```

2. ユーザーがプロビジョニングしたインフラストラクチャーを使用して、クラスターをデプロイする場合:

- Kubernetes マニフェストファイルを作成したら、**cluster-scheduler-02-config.yml** ファイルで **spec.mastersSchedulable** パラメーターが **true** に設定されていることを確認します。このファイルは、**<installation_directory>/manifests** にあります。詳細については、「CloudFormation テンプレートを使用して、AWS でユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールする」の「Kubernetes マニフェストと Ignition 設定ファイルの作成」を参照してください。
- 追加のワーカーノードを作成しないでください。

3 ノードクラスターの cluster-scheduler-02-config.yml ファイルの例

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  creationTimestamp: null
  name: cluster
spec:
  mastersSchedulable: true
  policy:
    name: ""
status: {}
```

6.5.2. 次のステップ

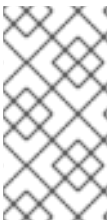
- カスタマイズによる AWS へのクラスターのインストール
- CloudFormation テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール

6.6. AWS でのクラスターのアンインストール

Amazon Web Services (AWS) にデプロイしたクラスターは削除することができます。

6.6.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。



注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

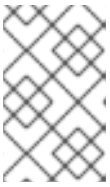
手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

6.6.2. Cloud Credential Operator ユーティリティーを使用した Amazon Web Services リソースの削除

クラスターの外部で管理される短期認証情報を使用する OpenShift Container Platform クラスターをアンインストールした後、CCO ユーティリティー (**ccoctl**) を使用して、インストール中に **ccoctl** が作成した Amazon Web Services (AWS) リソースを削除できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。
- 短期認証情報を使用する AWS 上の OpenShift Container Platform クラスターをアンインストールします。

手順

- 次のコマンドを実行して、**ccoctl** が作成した AWS リソースを削除します。

```
$ ccoctl aws delete \
  --name=<name> \1
  --region=<aws_region> 2
```

1 **<name>** は、クラウドリソースを最初に作成してタグ付けするために使用された名前と一致します。

2 **<aws_region>** は、クラウドリソースが削除される AWS リージョンです。

出力例

```
2021/04/08 17:50:41 Identity Provider object .well-known/openid-configuration deleted from
```

```
the bucket <name>-oidc
2021/04/08 17:50:42 Identity Provider object keys.json deleted from the bucket <name>-oidc
2021/04/08 17:50:43 Identity Provider bucket <name>-oidc deleted
2021/04/08 17:51:05 Policy <name>-openshift-cloud-credential-operator-cloud-credential-o
associated with IAM Role <name>-openshift-cloud-credential-operator-cloud-credential-o
deleted
2021/04/08 17:51:05 IAM Role <name>-openshift-cloud-credential-operator-cloud-credential-
o deleted
2021/04/08 17:51:07 Policy <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials
associated with IAM Role <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials deleted
2021/04/08 17:51:07 IAM Role <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials
deleted
2021/04/08 17:51:08 Policy <name>-openshift-image-registry-installer-cloud-credentials
associated with IAM Role <name>-openshift-image-registry-installer-cloud-credentials
deleted
2021/04/08 17:51:08 IAM Role <name>-openshift-image-registry-installer-cloud-credentials
deleted
2021/04/08 17:51:09 Policy <name>-openshift-ingress-operator-cloud-credentials associated
with IAM Role <name>-openshift-ingress-operator-cloud-credentials deleted
2021/04/08 17:51:10 IAM Role <name>-openshift-ingress-operator-cloud-credentials deleted
2021/04/08 17:51:11 Policy <name>-openshift-machine-api-aws-cloud-credentials associated
with IAM Role <name>-openshift-machine-api-aws-cloud-credentials deleted
2021/04/08 17:51:11 IAM Role <name>-openshift-machine-api-aws-cloud-credentials deleted
2021/04/08 17:51:39 Identity Provider with ARN arn:aws:iam::<aws_account_id>:oidc-
provider/<name>-oidc.s3.<aws_region>.amazonaws.com deleted
```

検証

- リソースが削除されたことを確認するには、AWS にクエリーを実行します。詳細は AWS ドキュメントを参照してください。

6.6.3. 設定された AWS Local Zone インフラストラクチャーを使用したクラスターの削除

Amazon Web Services (AWS) のクラスターを既存の Virtual Private Cloud (VPC) にインストールし、ローカルゾーンの場所ごとにサブネットを設定したら、クラスターとそれに関連付けられている AWS リソースを削除できます。

この手順の例では、CloudFormation テンプレートを使用して VPC とそのサブネットを作成したことを前提としています。

前提条件

- ネットワークの作成中に使用された CloudFormation スタックの名前 **<local_zone_stack_name>** と **<vpc_stack_name>** を知っています。クラスターを削除するには、スタックの名前が必要です。
- インストールプログラムによって作成されたインストールファイルを含むディレクトリーへのアクセス権があります。
- アカウントには、CloudFormation スタックを削除するためのアクセス許可を提供するポリシーが含まれています。

手順

1. インストールプログラムが保存されているディレクトリーに移動し、**destroy cluster** コマンドを使用してクラスターを削除します。

```
$ ./openshift-install destroy cluster --dir <installation_directory> \ 1
  --log-level=debug 2
```

1 **<installation_directory>** には、インストールプログラムによって作成されたファイルを保存したディレクトリーを指定します。

2 別のログの詳細を表示するには、**debug** の代わりに **error**、**info**、または **warn** を指定します。

2. Local Zone サブネットの CloudFormation スタックを削除します。

```
$ aws cloudformation delete-stack --stack-name <local_zone_stack_name>
```

3. VPC を表すリソースのスタックを削除します。

```
$ aws cloudformation delete-stack --stack-name <vpc_stack_name>
```

検証

- AWS CLI で次のコマンドを発行して、スタックリソースを削除したことを確認します。AWS CLI は、テンプレートコンポーネントが存在しないことを出力します。

```
$ aws cloudformation describe-stacks --stack-name <local_zone_stack_name>
```

```
$ aws cloudformation describe-stacks --stack-name <vpc_stack_name>
```

関連情報

- AWS CloudFormation スタックについての詳細は、Working with stacks を参照してください。
- AWS Local Zones へのオプトイン
- AWS Local Zones の利用可能なロケーション
- AWS Local Zones の機能

6.7. AWS のインストール設定パラメーター

OpenShift Container Platform クラスターを AWS にデプロイする前に、クラスターとそれをホストするプラットフォームをカスタマイズするためのパラメーターを指定します。**install-config.yaml** ファイルを作成するときは、コマンドラインを使用して必要なパラメーターの値を指定します。その後、**install-config.yaml** ファイルを変更して、クラスターをさらにカスタマイズできます。

6.7.1. AWS で使用可能なインストール設定パラメーター

次の表では、インストールプロセスの一部として設定できる、必須、オプション、および AWS 固有のインストール設定パラメーターを示します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

6.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表6.27 必須パラメーター

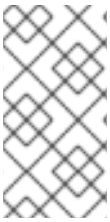
パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

6.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。



注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表6.28 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

パラメーター	説明	値
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

6.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表6.29 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列

パラメーター	説明	値
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** と **arm64** です。すべてのインストールオプションが 64 ビット ARM アーキテクチャーをサポートしているわけではありません。使用するインストールオプションがプラットフォームでサポートされているか確認するには、**クラスターインストール方法の選択およびそのユーザー向けの準備** の **各種プラットフォームでサポートされているインストール方法** 参照してください。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**

パラメーター	説明	値
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** と **arm64** です。すべてのインストールオプションが 64 ビット ARM アーキテクチャーをサポートしているわけではありません。使用するインストールオプションがプラットフォームでサポートされているか確認するには、**クラスターインストール方法の選択およびそのユーザー向けの準備** の **各種プラットフォームでサポートされているインストール方法** 参照してください。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。 **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**

パラメーター	説明	値
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]

パラメーター	説明	値
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
platform: aws: lbType:	AWS で NLB ロードバランサータイプを設定するために必要です。有効な値は **Classic** または **NLB** です。値が指定されていない場合、インストールプログラムはデフォルトで **Classic** になります。インストールプログラムは、ここで指定された値をイングレスクラスター設定オブジェクトに設定します。他の Ingress コントローラーのロードバランサータイプを指定しない場合、それらはこのパラメーターに設定されたタイプを使用します。	**Classic** または **NLB**デフォルト値は **Classic** です。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

注記

AWS アカウントでサービスコントロールポリシー (SCP) が有効になっている場合は、**credentialsMode** パラメーターを **Mint**、**Passthrough** または **Manual** に設定する必要があります。

重要

このパラメーターを **Manual** に設定すると、管理者レベルのシークレットを **kube-system** プロジェクトに保存する代替手段が有効になりますが、追加の設定手順が必要になります。詳細は、管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法を参照してください。

6.7.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

表6.30 オプションの AWS パラメーター

パラメーター	説明	値
compute: platform: aws: amiID:	クラスターのコンピュートマシンの起動に使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。利用可能な AMI ID については、**AWS インフラストラクチャーの RHCOS AMI** を参照してください。
compute: platform: aws: iamRole:	コンピュートマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
compute: platform: aws: rootVolume: iops:	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。

パラメーター	説明	値
compute: platform: aws: rootVolume: size:	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
compute: platform: aws: rootVolume: type:	root ボリュームのタイプです。	有効な AWS EBS ボリュームタイプ (例: **io1**)。
compute: platform: aws: rootVolume: kmsKeyARN:	KMS キーの Amazon リソース名 (キー ARN)。これは、ワーカーノードのオペレーティングシステムボリュームを特定の KMS キーで暗号化するために必要です。	有効なキー ID またはキー ARN。
compute: platform: aws: type:	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **m4.2xlarge**)。次の **サポートされている AWS マシンタイプの** 表を参照してください。
compute: platform: aws: zones:	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンのリスト。

パラメーター	説明	値
compute: aws: region:	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。AWS CLI を使用して、選択したインスタンスタイプに基づいて利用可能なリージョンにアクセスできます。以下に例を示します。 `aws ec2 describe-instance-type-offerings --filters Name=instance-type,Values=c7g.xlarge`  **重要** ARM ベースの AWS インスタンスで実行する場合は、AWS Graviton プロセッサーが利用可能なリージョンを入力するようにしてください。AWS ドキュメントの グローバルアベイラビリティー マップを参照してください。現在、AWS Graviton3 プロセッサーは一部のリージョンでのみ利用できます。
controlPlane: platform: aws: amiID:	クラスターのコントロールプレーンマシンを起動するために使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。利用可能な AMI ID については、**AWS インフラストラクチャーの RHCOS AMI** を参照してください。
controlPlane: platform: aws: iamRole:	コントロールプレーンマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。

パラメーター	説明	値
controlPlane: platform: aws: rootVolume: iops:	コントロールプレーンマシン上のルートボリューム用に予約されている 1 秒あたりの入出力操作数 (IOPS)。	整数 (例: **4000**)。
controlPlane: platform: aws: rootVolume: size:	コントロールプレーンマシンのルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
controlPlane: platform: aws: rootVolume: type:	コントロールプレーンマシンのルートボリュームのタイプ。	有効な AWS EBS ボリュームタイプ (例: **io1**)。
controlPlane: platform: aws: rootVolume: kmsKeyARN:	KMS キーの Amazon リソース名 (キー ARN)。これは、特定の KMS キーでコントロールプレーンノードのオペレーティングシステムボリュームを暗号化するために必要です。	有効なキー ID とキー ARN。
controlPlane: platform: aws: type:	コントロールプレーンマシンの EC2 インスタンスタイプ。	**m6i.xlarge** などの有効な AWS インスタンスタイプ。次の **サポートされている AWS マシンタイプの** 表を参照してください。
controlPlane: platform: aws: zones:	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンのリスト。

パラメーター	説明	値
controlPlane: aws: region:	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform: aws: amiID:	クラスターのすべてのマシンを起動するために使用される AWS AMI。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。利用可能な AMI ID については、**AWS インフラストラクチャーの RHCOS AMI** を参照してください。
platform: aws: hostedZone:	クラスターの既存の Route 53 プライベートホストゾーン。独自の VPC を指定する場合も、既存のホストゾーンのみを使用できます。ホストゾーンは、インストール前にユーザーによって提供される VPC に関連付けられている必要があります。また、ホストゾーンのドメインはクラスタードメインまたはクラスタードメインの親である必要があります。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。	文字列 (例: **Z3URY6TWQ91KVV**)
platform: aws: hostedZoneRole:	指定されたホストゾーンを含むアカウントの既存の IAM ロールの Amazon Resource Name (ARN)。インストールプログラムとクラスターオペレーターは、ホストゾーンで操作を実行するときにこのロールを引き受けます。このパラメーターは、クラスターを共有 VPC にインストールする場合にのみ使用してください。	文字列 (例 **arn:aws:iam::1234567890:role/shared-vpc-role**)。

パラメーター	説明	値
platform: aws: serviceEndpoints: - name: url:	AWS サービスのエンドポイント名と URL。カスタムエンドポイントは、FIPS などの AWS の代替エンドポイントを使用しなければならない場合にのみ必要です。カスタム API エンドポイントは、EC2、S3、IAM、Elastic Load Balancing、Tagging、Route 53、および STS AWS サービスに指定できます。	有効な AWS サービスエンドポイント 名と有効な AWS サービスエンドポイント URL。
platform: aws: userTags:	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。 **注記** インストール時に、最大 25 個のユーザー定義タグを追加できます。残りの 25 個のタグは、OpenShift Container Platform 用に予約されています。
platform: aws: propagateUserTags:	クラスター内 Operator に対し、Operator が作成する AWS リソースのタグに指定されたユーザータグを組み込むフラグ。	ブール値（**true** または **false** など）。

パラメーター	説明	値
platform: aws: subnets:	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。 標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。 プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。 AWS Local Zone を使用するクラスターの場合、エッジマシンプールが確実に作成されるように、AWS Local Zone のサブネットをこのリストに追加する必要があります。	有効なサブネット ID。
platform: aws: publicIpv4Pool: 	**publish** が **External** に設定されている場合に Elastic IP (EIP)の割り当てに使用されるパブリック IPv4 プール ID。クラスターの同じ AWS アカウントおよびリージョンでプールをプロビジョニングおよびアドバタイズする必要があります。プールに 2n + 1 IPv4 が利用可能であることを確認する必要があります。**n** は、API、NAT ゲートウェイ、およびブートストラップノードの Network Load Balancer (NLB)をデプロイするために使用される AWS ゾーンの合計数です。独自の IP アドレス(BYOIP)を AWS に取り込む方法の詳細については、Onboard your BYOIP を参照してください。	有効な パブリック IPv4 プール ID **注記** BYOIP は、ネットワークの制限のないカスタマイズされたインストールに対してのみ有効にできます。

パラメーター	説明	値
platform: aws: preserveBootstrapIgnition:	ブートストラップの完了後に S3 バケットが削除されないようにします。	**true** または **false**。デフォルト値は **false** で、S3 バケットが削除されます。

第7章 AZURE へのインストール

7.1. AZURE へのインストールの準備

7.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。

7.1.2. OpenShift Container Platform の Azure へのインストール要件

OpenShift Container Platform を Microsoft Azure にインストールする前に、Azure アカウントを設定する必要があります。アカウントの設定、アカウントの制限、パブリック DNS ゾーン設定、必要なロール、サービスプリンシパルの作成、およびサポートされる Azure リージョンの詳細は、Azure アカウントの設定 を参照してください。

ご使用の環境でクラウド ID およびアクセス管理 (IAM) API にアクセスできない場合、または管理者レベルの認証情報シークレットを **kube-system** namespace に保存しない場合は、管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法 で他のオプションを参照してください。

7.1.3. Azure に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーによるプロビジョニングおよびユーザーによるプロビジョニングのインストールプロセスの詳細は、インストールプロセス を参照してください。

7.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる Azure インフラストラクチャーに、クラスターをインストールできます。

- **クラスターの Azure へのクイックインストール**: OpenShift Container Platform インストールプログラムでプロビジョニングされる Azure インフラストラクチャーに OpenShift Container Platform をインストールできます。デフォルトの設定オプションを使用して、クラスターを迅速にインストールできます。
- **カスタマイズされたクラスターの Azure へのインストール**: インストールプログラムがプロビジョニングする Azure インフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の多くのカスタマイズオプションは、インストール後 に利用できます。
- **ネットワークのカスタマイズを使用したクラスターの Azure へのインストール**: インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。

- **Azure の既存 VNet へのクラスターのインストール**: OpenShift Container Platform を Azure の既存の Azure Virtual Network (VNet) にインストールできます。このインストール方法は、新規アカウントまたはインフラストラクチャーを作成する際の制限など、会社のガイドラインによる制約がある場合に使用できます。

- **プライベートクラスターの Azure へのインストール**: プライベートクラスターを Azure の既存の Azure Virtual Network (VNet) にインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

- **Azure の government リージョンへのクラスターのインストール**: OpenShift Container Platform は、機密ワークロードを Azure で実行する必要のある連邦、州、地方の米国の各種の政府機関、請負業者、教育機関、およびその他の米国の顧客向けに設計されている Microsoft Azure Government (MAG) リージョンにデプロイできます。

7.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法を使用して、独自にプロビジョニングする Azure インフラストラクチャーにクラスターをインストールできます。

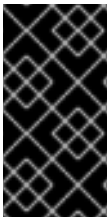
- **ARM テンプレートを使用したクラスターの Azure へのインストール**: 独自に提供するインフラストラクチャーを使用して、OpenShift Container Platform を Azure にインストールできます。提供される Azure Resource Manager (ARM) テンプレートを使用して、インストールを支援できます。

7.1.4. 次のステップ

- Azure アカウントの設定

7.2. AZURE アカウントの設定

OpenShift Container Platform をインストールする前に、インストール要件を満たすように Microsoft Azure アカウントを設定する必要があります。



重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

7.2.1. Azure アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure コンポーネントを使用し、デフォルトの Azure サブスクリプションおよびサービス制限、クォータ、および制約 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。

重要

デフォルトの制限は、Free Trial や Pay-As-You-Go、および DV2、F、および G などのシリーズといったカテゴリータイプによって異なります。たとえば、Enterprise Agreement サブスクリプションのデフォルトは 350 コアです。

サブスクリプションタイプの制限を確認し、必要に応じて、デフォルトのクラスターを Azure にインストールする前にアカウントのクォータ制限を引き上げます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
vCPU	44	リージョンごとに 20	デフォルトのクラスターには 44 vCPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップおよびコントロールプレーンマシンは 8 vCPU を使用する **Standard_D8s_v3** 仮想マシンを使用し、コンピュートマシンは 4 vCPU を使用する **Standard_D4s_v3** 仮想マシンを使用するため、 デフォルトのクラスターには 44 vCPU が必要です。8 vCPU を使用するブートストラップノードの仮想マシンは、インストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。
OS ディスク	7		各クラスターマシンには、少なくとも 100 GB のストレージと 300 IOPS が必要です。これらはサポートされる最小の値ですが、実稼働クラスターおよび高負荷ワークロードがあるクラスターには、さらに高速なストレージが推奨されます。パフォーマンスを向上させるためのストレージ最適化について、詳しくは「スケーラビリティーとパフォーマンス」セクションの「ストレージの最適化」を参照してください。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
VNet	1	リージョンごとに 1000	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。
ネットワークインターフェイス	7	リージョンごとに 65,536	各デフォルトクラスターには、7 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。
ネットワークセキュリティーグループ	2	5000	各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。
ネットワークロードバランサー	3	リージョンごとに 1000	各クラスターは以下の ロードバランサー を作成します。 **default**: ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス **internal**: コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス **external**: コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス アプリケーションが追加の Kubernetes **LoadBalancer** サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
パブリック IP アドレス	3		2 つのパブリックロードバランサーのそれぞれはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。
プライベート IP アドレス	7		内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。
スポット VM vCPU (オプション)	0 スポット VM を設定する場合には、クラスターのコンピュートノードごとにスポット VM vCPU が 2 つ必要です。	リージョンごとに 20	これはオプションのコンポーネントです。スポット VM を使用するには、Azure のデフォルトの制限を最低でも、クラスター内のコンピュートノード数の 2 倍に増やす必要があります。  **注記** コントロールプレーンノードにスポット VM を使用することは推奨しません。

関連情報

- ストレージの最適化

7.2.2. Azure でのパブリック DNS ゾーンの設定

OpenShift Container Platform をインストールするには、使用する Microsoft Azure アカウントに、専用のパブリックホスト DNS ゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。このサービスは、クラスターへの外部接続のためのクラスター DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、Azure または別のソースから新規のものを取得できます。

注記

Azure 経由でドメインを購入する方法についての詳細は、Azure ドキュメントの Buy a custom domain name for Azure App Service を参照してください。

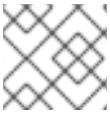
2. 既存のドメインおよびレジストラーを使用している場合、その DNS を Azure に移行します。Azure ドキュメントの Migrate an active DNS name to Azure App Service を参照してください。

3. ドメインの DNS を設定します。Azure ドキュメントの Tutorial: Host your domain in Azure DNS の手順に従い、ドメインまたはサブドメインのパブリックホストゾーンを作成し、 新規の権威ネームサーバーを抽出し、ドメインが使用するネームサーバーのレジストラーレコードを更新します。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。

7.2.3. Azure アカウント制限の拡張

アカウントの制限を引き上げるには、Azure ポータルでサポートをリクエストします。



注記

サポートリクエストごとに 1 つの種類のクォータのみを増やすことができます。

手順

1. Azure ポータルの左端で **Help + support** をクリックします。
2. **New support request** をクリックしてから必要な値を選択します。
    a. **Issue type** リストから、**Service and subscription limits (quotas)** を選択します。
    b. **Subscription** リストから、変更するサブスクリプションを選択します。
    c. **Quota type** リストから、引き上げるクォータを選択します。たとえば、**Compute-VM (cores-vCPUs) subscription limit increases** を選択し、クラスターのインストールに必要な vCPU の数を増やします。
    d. **Next: Solutions** をクリックします。
3. **Problem Details**ページで、クォータの引き上げについての必要な情報を指定します。
    a. **Provide details**をクリックし、**Quota details**ウィンドウに必要な詳細情報を指定します。
    b. SUPPORT METHOD and CONTACT INFO セクションに、問題の重大度および問い合わせ先の詳細を指定します。
4. **Next: Review + create** をクリックしてから **Create** をクリックします。

7.2.4. サブスクリプション ID とテナント ID の記録

インストールプログラムには、Azure アカウントに関連付けられたサブスクリプション ID とテナント ID が必要です。Azure CLI を使用してこの情報を収集できます。

前提条件

- Azure CLI をインストールまたは更新している。

手順

1. 次のコマンドを実行して、Azure CLI にログインします。

```
$ az login
```

2. 適切なサブスクリプションを使用していることを確認してください。

   a. 次のコマンドを実行して、利用可能なサブスクリプションのリストを表示します。

```
$ az account list --refresh
```

   出力例

```
[
  {
    "cloudName": "AzureCloud",
    "id": "8xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "isDefault": true,
    "name": "Subscription Name 1",
    "state": "Enabled",
    "tenantId": "6xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  },
  {
    "cloudName": "AzureCloud",
    "id": "9xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "isDefault": false,
    "name": "Subscription Name 2",
    "state": "Enabled",
    "tenantId": "7xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "user": {
      "name": "you2@example.com",
      "type": "user"
    }
  }
]
```

   b. 次のコマンドを実行して、アクティブなアカウントの詳細を表示し、これが使用するサブスクリプションであることを確認します。

```
$ az account show
```

   出力例

```
{
  "environmentName": "AzureCloud",
  "id": "8xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "isDefault": true,
  "name": "Subscription Name 1",
  "state": "Enabled",
```

```
  "tenantId": "6xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 適切なサブスクリプションを使用していない場合は、以下を行います。

   a. 次のコマンドを実行して、アクティブなサブスクリプションを変更します。

   ```
   $ az account set -s <subscription_id>
   ```

   b. 次のコマンドを実行して、必要なサブスクリプションを使用していることを確認します。

   ```
   $ az account show
   ```

   出力例

   ```
   {
     "environmentName": "AzureCloud",
     "id": "9xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
     "isDefault": true,
     "name": "Subscription Name 2",
     "state": "Enabled",
     "tenantId": "7xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
     "user": {
       "name": "you2@example.com",
       "type": "user"
     }
   }
   ```

4. 出力から **id** および **tenantId** パラメーターの値を記録します。OpenShift Container Platform クラスターをインストールするには、これらの値が必要です。

7.2.5. Azure リソースにアクセスするためにサポートされている ID

OpenShift Container Platform クラスターでは、Azure リソースを作成および管理するために Azure ID が必要です。したがって、インストールを完了するには、次のいずれかのタイプの ID が必要です。

- サービスプリンシパル
- システムが割り当てたマネージド ID
- ユーザーが割り当てたマネージド ID

7.2.5.1. 必要な Azure ロール

OpenShift Container Platform クラスターでは、Azure リソースを作成および管理するために Azure ID が必要です。ID を作成する前に、環境が次の要件を満たしていることを確認してください。

- ID の作成に使用する Azure アカウントには、**User Access Administrator** ロールと **Contributor** ロールが割り当てられます。これらのロールは次の場合に必要です。

- サービスプリンシパルまたはユーザー割り当てのマネージド ID を作成します。
- 仮想マシン上でシステム割り当てマネージド ID を有効にします。

- サービスプリンシパルを使用してインストールを完了する場合は、ID の作成に使用する Azure アカウントに、Microsoft Entra ID の **Microsoft.directory/servicePrincipals/createAsOwner** パーミッションが割り当てられていることを確認してください。

Azure ポータルでロールを設定するには、Azure ドキュメントの Manage access to Azure resources using RBAC and the Azure portal を参照します。

7.2.5.2. インストーラーでプロビジョニングされるインフラストラクチャーのインストールサポート用の BYOH

インストールプログラムでは、クラスターをデプロイし、日常の操作を維持するために必要なパーミッションを持つ Azure サービスプリンシパルまたはマネージド ID にアクセスする必要があります。これらパーミッションは、ID に関連付けられた Azure サブスクリプションに付与する必要があります。

以下のオプションを使用できます。

- ID に **Contributor** ロールと **User Access Administrator** ロールを割り当てることができます。これらのロールを割り当てるのが、必要な権限をすべて付与する最も簡単な方法です。
ロールの割り当ての詳細は、Azure portal を使用した Azure リソースへのアクセスの管理 に関する Azure ドキュメントを参照してください。

- 組織のセキュリティーポリシーで、より制限的なアクセス許可のセットが必要な場合は、必要なアクセス許可を持つ カスタムロール を作成できます。

Microsoft Azure で OpenShift Container Platform クラスターを作成するには、以下のアクセス許可が必要です。

> **例7.1 承認リソースを作成するために必要な権限**
>
> - **Microsoft.Authorization/policies/audit/action**
> - **Microsoft.Authorization/policies/auditIfNotExists/action**
> - **Microsoft.Authorization/roleAssignments/read**
> - **Microsoft.Authorization/roleAssignments/write**

> **例7.2 コンピューティングリソースの作成に必要な権限**
>
> - **Microsoft.Compute/availabilitySets/write**
> - **Microsoft.Compute/availabilitySets/read**
> - **Microsoft.Compute/disks/beginGetAccess/action**
> - **Microsoft.Compute/disks/delete**
> - **Microsoft.Compute/disks/read**
> - **Microsoft.Compute/disks/write**

- **Microsoft.Compute/galleries/images/read**
- **Microsoft.Compute/galleries/images/versions/read**
- **Microsoft.Compute/galleries/images/versions/write**
- **Microsoft.Compute/galleries/images/write**
- **Microsoft.Compute/galleries/read**
- **Microsoft.Compute/galleries/write**
- **Microsoft.Compute/snapshots/read**
- **Microsoft.Compute/snapshots/write**
- **Microsoft.Compute/snapshots/delete**
- **Microsoft.Compute/virtualMachines/delete**
- **Microsoft.Compute/virtualMachines/powerOff/action**
- **Microsoft.Compute/virtualMachines/read**
- **Microsoft.Compute/virtualMachines/write**

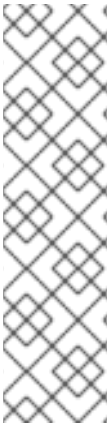
例7.3 ID 管理リソースを作成するために必要なアクセス許可

- **Microsoft.ManagedIdentity/userAssignedIdentities/assign/action**
- **Microsoft.ManagedIdentity/userAssignedIdentities/read**
- **Microsoft.ManagedIdentity/userAssignedIdentities/write**

例7.4 ネットワークリソースの作成に必要な権限

- **Microsoft.Network/dnsZones/A/write**
- **Microsoft.Network/dnsZones/CNAME/write**
- **Microsoft.Network/dnszones/CNAME/read**
- **Microsoft.Network/dnszones/read**
- **Microsoft.Network/loadBalancers/backendAddressPools/join/action**
- **Microsoft.Network/loadBalancers/backendAddressPools/read**
- **Microsoft.Network/loadBalancers/backendAddressPools/write**
- **Microsoft.Network/loadBalancers/read**
- **Microsoft.Network/loadBalancers/write**
- **Microsoft.Network/networkInterfaces/delete**

- **Microsoft.Network/networkInterfaces/join/action**
- **Microsoft.Network/networkInterfaces/read**
- **Microsoft.Network/networkInterfaces/write**
- **Microsoft.Network/networkSecurityGroups/join/action**
- **Microsoft.Network/networkSecurityGroups/read**
- **Microsoft.Network/networkSecurityGroups/securityRules/delete**
- **Microsoft.Network/networkSecurityGroups/securityRules/read**
- **Microsoft.Network/networkSecurityGroups/securityRules/write**
- **Microsoft.Network/networkSecurityGroups/write**
- **Microsoft.Network/privateDnsZones/A/read**
- **Microsoft.Network/privateDnsZones/A/write**
- **Microsoft.Network/privateDnsZones/A/delete**
- **Microsoft.Network/privateDnsZones/SOA/read**
- **Microsoft.Network/privateDnsZones/read**
- **Microsoft.Network/privateDnsZones/virtualNetworkLinks/read**
- **Microsoft.Network/privateDnsZones/virtualNetworkLinks/write**
- **Microsoft.Network/privateDnsZones/write**
- **Microsoft.Network/publicIPAddresses/delete**
- **Microsoft.Network/publicIPAddresses/join/action**
- **Microsoft.Network/publicIPAddresses/read**
- **Microsoft.Network/publicIPAddresses/write**
- **Microsoft.Network/virtualNetworks/join/action**
- **Microsoft.Network/virtualNetworks/read**
- **Microsoft.Network/virtualNetworks/subnets/join/action**
- **Microsoft.Network/virtualNetworks/subnets/read**
- **Microsoft.Network/virtualNetworks/subnets/write**
- **Microsoft.Network/virtualNetworks/write**



注記

Azure でプライベート OpenShift Container Platform クラスターを作成するために、以下のアクセス許可は必要ありません。

- **Microsoft.Network/dnsZones/A/write**
- **Microsoft.Network/dnsZones/CNAME/write**
- **Microsoft.Network/dnszones/CNAME/read**
- **Microsoft.Network/dnszones/read**

例7.5 リソースの正常性をチェックするために必要なアクセス許可

- **Microsoft.Resourcehealth/healthevent/Activated/action**
- **Microsoft.Resourcehealth/healthevent/InProgress/action**
- **Microsoft.Resourcehealth/healthevent/Pending/action**
- **Microsoft.Resourcehealth/healthevent/Resolved/action**
- **Microsoft.Resourcehealth/healthevent/Updated/action**

例7.6 リソースグループの作成に必要なアクセス許可

- **Microsoft.Resources/subscriptions/resourceGroups/read**
- **Microsoft.Resources/subscriptions/resourcegroups/write**

例7.7 リソースタグの作成に必要なアクセス許可

- **Microsoft.Resources/tags/write**

例7.8 ストレージリソースの作成に必要な権限

- **Microsoft.Storage/storageAccounts/blobServices/read**
- **Microsoft.Storage/storageAccounts/blobServices/containers/write**
- **Microsoft.Storage/storageAccounts/fileServices/read**
- **Microsoft.Storage/storageAccounts/fileServices/shares/read**
- **Microsoft.Storage/storageAccounts/fileServices/shares/write**
- **Microsoft.Storage/storageAccounts/fileServices/shares/delete**
- **Microsoft.Storage/storageAccounts/listKeys/action**
- **Microsoft.Storage/storageAccounts/read**

- **Microsoft.Storage/storageAccounts/write**

例7.9 イメージレジストリーのプライベートストレージエンドポイントの作成に使用するオプションの権限

- **Microsoft.Network/privateEndpoints/write**
- **Microsoft.Network/privateEndpoints/read**
- **Microsoft.Network/privateEndpoints/privateDnsZoneGroups/write**
- **Microsoft.Network/privateEndpoints/privateDnsZoneGroups/read**
- **Microsoft.Network/privateDnsZones/join/action**
- **Microsoft.Storage/storageAccounts/PrivateEndpointConnectionsApproval/action**

例7.10 Marketplace 仮想マシンリソースを作成するためのオプションのアクセス許可

- **Microsoft.MarketplaceOrdering/offertypes/publishers/offers/plans/agreements/read**
- **Microsoft.MarketplaceOrdering/offertypes/publishers/offers/plans/agreements/write**

例7.11 コンピュートリソースを作成するためのオプションのアクセス許可

- **Microsoft.Compute/images/read**
- **Microsoft.Compute/images/write**
- **Microsoft.Compute/images/delete**

例7.12 ユーザー管理の暗号化を有効にするためのオプションのアクセス許可

- **Microsoft.Compute/diskEncryptionSets/read**
- **Microsoft.Compute/diskEncryptionSets/write**
- **Microsoft.Compute/diskEncryptionSets/delete**
- **Microsoft.KeyVault/vaults/read**
- **Microsoft.KeyVault/vaults/write**
- **Microsoft.KeyVault/vaults/delete**
- **Microsoft.KeyVault/vaults/deploy/action**
- **Microsoft.KeyVault/vaults/keys/read**
- **Microsoft.KeyVault/vaults/keys/write**
- **Microsoft.Features/providers/features/register/action**

例7.13 NatGateway アウトバウンドタイプを使用してクラスターをインストールするためのオプションの権限

- **Microsoft.Network/natGateways/read**
- **Microsoft.Network/natGateways/write**

例7.14 Azure ネットワークアドレス変換 (NAT) を使用してプライベートクラスターをインストールするためのオプションのアクセス許可

- **Microsoft.Network/natGateways/join/action**
- **Microsoft.Network/natGateways/read**
- **Microsoft.Network/natGateways/write**

例7.15 Azure ファイアウォールを使用してプライベートクラスターをインストールするためのオプションのアクセス許可

- **Microsoft.Network/azureFirewalls/applicationRuleCollections/write**
- **Microsoft.Network/azureFirewalls/read**
- **Microsoft.Network/azureFirewalls/write**
- **Microsoft.Network/routeTables/join/action**
- **Microsoft.Network/routeTables/read**
- **Microsoft.Network/routeTables/routes/read**
- **Microsoft.Network/routeTables/routes/write**
- **Microsoft.Network/routeTables/write**
- **Microsoft.Network/virtualNetworks/peer/action**
- **Microsoft.Network/virtualNetworks/virtualNetworkPeerings/read**
- **Microsoft.Network/virtualNetworks/virtualNetworkPeerings/write**

例7.16 収集ブートストラップを実行するためのオプションの権限

- **Microsoft.Compute/virtualMachines/instanceView/read**

Microsoft Azure で OpenShift Container Platform クラスターを削除するには、以下のアクセス許可が必要です。同じアクセス許可を使用して、Azure 上のプライベート OpenShift Container Platform クラスターを削除できます。

例7.17 承認リソースを削除するために必要な権限

- **Microsoft.Authorization/roleAssignments/delete**

例7.18 コンピューティングリソースを削除するために必要な権限

- **Microsoft.Compute/disks/delete**
- **Microsoft.Compute/galleries/delete**
- **Microsoft.Compute/galleries/images/delete**
- **Microsoft.Compute/galleries/images/versions/delete**
- **Microsoft.Compute/virtualMachines/delete**

例7.19 Required permissions for deleting identity management resources

- **Microsoft.ManagedIdentity/userAssignedIdentities/delete**

例7.20 ネットワークリソースを削除するために必要な権限

- **Microsoft.Network/dnszones/read**
- **Microsoft.Network/dnsZones/A/read**
- **Microsoft.Network/dnsZones/A/delete**
- **Microsoft.Network/dnsZones/CNAME/read**
- **Microsoft.Network/dnsZones/CNAME/delete**
- **Microsoft.Network/loadBalancers/delete**
- **Microsoft.Network/networkInterfaces/delete**
- **Microsoft.Network/networkSecurityGroups/delete**
- **Microsoft.Network/privateDnsZones/read**
- **Microsoft.Network/privateDnsZones/A/read**
- **Microsoft.Network/privateDnsZones/delete**
- **Microsoft.Network/privateDnsZones/virtualNetworkLinks/delete**
- **Microsoft.Network/publicIPAddresses/delete**
- **Microsoft.Network/virtualNetworks/delete**

注記

Azure 上のプライベート OpenShift Container Platform クラスターを削除するために、以下のアクセス許可は必要ありません。

- **Microsoft.Network/dnszones/read**
- **Microsoft.Network/dnsZones/A/read**
- **Microsoft.Network/dnsZones/A/delete**
- **Microsoft.Network/dnsZones/CNAME/read**
- **Microsoft.Network/dnsZones/CNAME/delete**

例7.21 リソースの正常性をチェックするために必要なアクセス許可

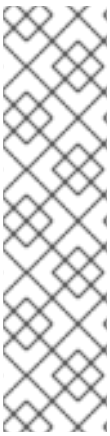
- **Microsoft.Resourcehealth/healthevent/Activated/action**
- **Microsoft.Resourcehealth/healthevent/Resolved/action**
- **Microsoft.Resourcehealth/healthevent/Updated/action**

例7.22 リソースグループを削除するために必要なアクセス許可

- **Microsoft.Resources/subscriptions/resourcegroups/delete**

例7.23 ストレージリソースを削除するために必要な権限

- **Microsoft.Storage/storageAccounts/delete**
- **Microsoft.Storage/storageAccounts/listKeys/action**



注記

Azure に OpenShift Container Platform をインストールするには、アクセス許可の範囲をサブスクリプションに限定する必要があります。後で、これらのアクセス許可の範囲を、インストーラーによって作成されたリソースグループに再設定できます。パブリック DNS ゾーンが別のリソースグループに存在する場合は、ネットワーク DNS ゾーンに関連するアクセス許可を常にサブスクリプションに適用する必要があります。デフォルトでは、OpenShift Container Platform インストールプログラムは Azure ID に **Contributor** ロールを割り当てます。

OpenShift Container Platform クラスターを削除するときに、すべてのパーミッションをサブスクリプションに限定できます。

7.2.5.3. Azure マネージド ID の使用

インストールプログラムでは、インストールを完了するために Azure ID が必要です。システム割り当てまたはユーザー割り当てのマネージド ID を使用できます。

マネージド ID を使用できない場合は、サービスプリンシパルを使用できます。

手順

1. システム割り当てのマネージド ID を使用している場合は、インストールプログラムを実行する仮想マシン上でそれを有効にします。
2. ユーザーが割り当てたマネージド ID を使用している場合は以下を行います。
   a. これを、インストールプログラムを実行する仮想マシンに割り当てます。
   b. クライアント ID を記録します。この値は、クラスターをインストールするときに必要になります。
   ユーザー割り当てマネージド ID の詳細を表示する場合については、Microsoft Azure ドキュメントで ユーザー割り当てマネージド ID のリスト を参照してください。
3. 必要なパーミッションがマネージド ID に割り当てられていることを確認します。

7.2.5.4. サービスプリンシパルの作成

インストールプログラムでは、インストールを完了するために Azure ID が必要です。サービスプリンシパルを使用できます。

サービスプリンシパルを使用できない場合は、マネージド ID を使用できます。

前提条件

- Azure CLI をインストールまたは更新している。
- Azure サブスクリプション ID がある。
- サービスプリンシパルに **Contributor** ロールおよび **User Administrator Access** ロールを割り当てない場合は、必要な Azure アクセス許可を持つカスタムロールを作成しています。

手順

1. 次のコマンドを実行して、アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role <role_name> \ 1
    --name <service_principal> \ 2
    --scopes /subscriptions/<subscription_id> 3
```

1. ロール名を定義します。**Contributor** ロールを使用するか、必要なアクセス許可を含むカスタムロールを指定できます。
2. サービスプリンシパル名を定義します。
3. サブスクリプション ID を指定します。

出力例

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
```

```
  "appId": "axxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "displayName": <service_principal>",
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

2. 出力から **appId** パラメーターと **password** パラメーターの値を記録します。クラスターをインストールするときにこれらの値が必要です。

3. **Contributor** ロールをサービスプリンシパルに適用した場合は、次のコマンドを実行して **User Administrator Access** ロールを割り当てます。

```
$ az role assignment create --role "User Access Administrator" \
  --assignee-object-id $(az ad sp show --id <appId> --query id -o tsv) 1
  --scope /subscriptions/<subscription_id> 2
```

1 サービスプリンシパルの **appId** パラメーター値を指定します。

2 サブスクリプション ID を指定します。

関連情報

- Cloud Credential Operator について

7.2.6. サポートされている Azure Marketplace リージョン

北米および EMEA でオファーを購入したお客様は、Azure Marketplace イメージを使用してクラスターをインストールすることができます。

このオファーは北米または EMEA で購入する必要がありますが、OpenShift Container Platform がサポートする任意の Azure パブリックパーティションにクラスターをデプロイできます。

注記

Azure Marketplace イメージを使用したクラスターのデプロイは、Azure Government リージョンではサポートされていません。

7.2.7. サポート対象の Azure リージョン

インストールプログラムは、サブスクリプションに基づいて利用可能な Microsoft Azure リージョンのリストを動的に生成します。

サポート対象の Azure パブリックリージョン

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)
- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)

- **canadaeast** (Canada East)
- **centralindia** (Central India)
- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)
- **eastus2** (East US 2)
- **francecentral** (France Central)
- **germanywestcentral** (Germany West Central)
- **israelcentral** (イスラエル中央)
- **italynorth** (イタリア北部)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **mexicocentral** (Mexico Central)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)
- **norwayeast** (Norway East)
- **polandcentral** (ポーランド中央)
- **qarcentral** (カタール中部)
- **southafricanorth** (South Africa North)
- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)
- **swedencentral** (スウェーデン中央)
- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)
- **uksouth** (UK South)
- **ukwest** (UK West)

- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)
- **westus3**(West US 3)

サポート対象の Azure Government リージョン
以下の Microsoft Azure Government (MAG) リージョンのサポートが OpenShift Container Platform バージョン 4.6 に追加されています。

- **usgovtexas** (US Gov Texas)
- **usgovvirginia** (US Gov Virginia)

Azure ドキュメント の利用可能なすべての MAG リージョンを参照できます。他の提供される MAG リージョンは OpenShift Container Platform で機能することが予想されますが、まだテストされていません。

7.2.8. 次のステップ

- OpenShift Container Platform クラスターを Azure にインストールします。カスタマイズされたクラスターのインストール 、またはデフォルトのオプションで クラスターのクイックインストール を実行できます。

7.3. AZURE のユーザー管理暗号化を有効にする

OpenShift Container Platform バージョン 4.16 では、ユーザー管理の暗号鍵を使用して Azure にクラスターをインストールできます。この機能を有効にするには、インストール前に、Azure DiskEncryptionSet を準備し、**install-config.yaml** ファイルを変更し、インストールを完了します。

7.3.1. Azure ディスク暗号化セットの準備

OpenShift Container Platform インストーラーは、ユーザー管理のキーで既存のディスク暗号化セットを使用できます。この機能を有効にするには、Azure でディスク暗号化セットを作成し、インストーラーにキーを提供します。

手順

1. 次のコマンドを実行して、Azure リソースグループの次の環境変数を設定します。

```
$ export RESOURCEGROUP="<resource_group>" \ 1
  LOCATION="<location>" 2
```

1. ディスク暗号化セットと暗号化キーを作成する Azure リソースグループの名前を指定します。クラスターを破棄した後にキーへのアクセスが失われないようにするには、クラスターをインストールするリソースグループとは別のリソースグループにディスク暗号化セットを作成する必要があります。
2. リソースグループを作成する Azure の場所を指定します。

2. 次のコマンドを実行して、Azure Key Vault とディスク暗号化セットの次の環境変数を設定します。

```
$ export KEYVAULT_NAME="<keyvault_name>" \ 1
    KEYVAULT_KEY_NAME="<keyvault_key_name>" \ 2
    DISK_ENCRYPTION_SET_NAME="<disk_encryption_set_name>" 3
```

1. 作成する Azure Key Vault の名前を指定します。
2. 作成する暗号化キーの名前を指定します。
3. 作成するディスク暗号化セットの名前を指定します。

3. 次のコマンドを実行して、Azure サービスプリンシパルの ID の環境変数を設定します。

```
$ export CLUSTER_SP_ID="<service_principal_id>" 1
```

1. このインストールに使用するサービスプリンシパルの ID を指定します。

4. 次のコマンドを実行して、Azure でホストレベルの暗号化を有効にします。

```
$ az feature register --namespace "Microsoft.Compute" --name "EncryptionAtHost"
```

```
$ az feature show --namespace Microsoft.Compute --name EncryptionAtHost
```

```
$ az provider register -n Microsoft.Compute
```

5. 次のコマンドを実行して、ディスク暗号化セットと関連リソースを保持する Azure リソースグループを作成します。

```
$ az group create --name $RESOURCEGROUP --location $LOCATION
```

6. 次のコマンドを実行して、Azure キー vault を作成します。

```
$ az keyvault create -n $KEYVAULT_NAME -g $RESOURCEGROUP -l $LOCATION \
    --enable-purge-protection true
```

7. 次のコマンドを実行して、キー vault に暗号化キーを作成します。

```
$ az keyvault key create --vault-name $KEYVAULT_NAME -n $KEYVAULT_KEY_NAME \
    --protection software
```

8. 次のコマンドを実行して、キー vault の ID をキャプチャーします。

```
$ KEYVAULT_ID=$(az keyvault show --name $KEYVAULT_NAME --query "[id]" -o tsv)
```

9. 次のコマンドを実行して、キー vault 内のキー URL をキャプチャーします。

```
$ KEYVAULT_KEY_URL=$(az keyvault key show --vault-name $KEYVAULT_NAME --name
\
  $KEYVAULT_KEY_NAME --query "[key.kid]" -o tsv)
```

10. 次のコマンドを実行して、ディスク暗号化セットを作成します。

```
$ az disk-encryption-set create -n $DISK_ENCRYPTION_SET_NAME -l $LOCATION -g \
  $RESOURCEGROUP --source-vault $KEYVAULT_ID --key-url $KEYVAULT_KEY_URL
```

11. 次のコマンドを実行して、キー vault へのアクセス権を DiskEncryptionSet リソースに付与します。

```
$ DES_IDENTITY=$(az disk-encryption-set show -n $DISK_ENCRYPTION_SET_NAME -g \
  $RESOURCEGROUP --query "[identity.principalId]" -o tsv)
```

```
$ az keyvault set-policy -n $KEYVAULT_NAME -g $RESOURCEGROUP --object-id \
  $DES_IDENTITY --key-permissions wrapkey unwrapkey get
```

12. 次のコマンドを実行して、Azure サービスプリンシパルに DiskEncryptionSet を読み取るパーミッションを付与します。

```
$ DES_RESOURCE_ID=$(az disk-encryption-set show -n
$DISK_ENCRYPTION_SET_NAME -g \
  $RESOURCEGROUP --query "[id]" -o tsv)
```

```
$ az role assignment create --assignee $CLUSTER_SP_ID --role "<reader_role>" \ 1
  --scope $DES_RESOURCE_ID -o jsonc
```

1 ディスク暗号化セットへの読み取りパーミッションを持つ Azure ロールを指定します。必要なアクセス許可を持つ **所有者** ロールまたはカスタムロールを使用できます。

7.3.2. 次のステップ

- OpenShift Container Platform クラスターをインストールします。
  - インストーラーでプロビジョニングされるインフラストラクチャーへのカスタマイズを使用したクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのネットワークのカスタマイズを使用したクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーでの既存の VNet へのクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのプライベートクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーでの government リージョンへのクラスターのインストール

7.4. クラスターの AZURE へのクイックインストール

OpenShift Container Platform バージョン 4.16 では、デフォルトの設定オプションを使用するクラスターを Microsoft Azure にインストールできます。

7.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

7.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.4.3. クラスターノードの SSH アクセス用のキーペアの生成

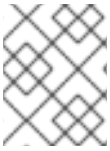
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

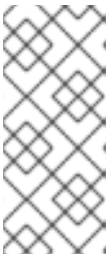
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

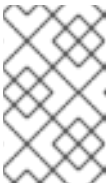
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

(1) ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.4.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

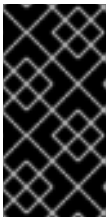
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.4.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルのアプリケーション ID とパスワードがある。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルを使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。

2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

3. プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **azure** を選択します。
インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。

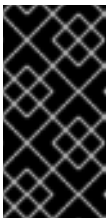
c. サブスクリプションとサービスプリンシパルに対して次の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
- **azure tenant id** テナント ID を入力します。
- **azure サービスプリンシパルクライアント ID**: アプリケーション ID を入力します。
- **azure サービスプリンシパルクライアントシークレット**: パスワードを入力します。

d. クラスターをデプロイするリージョンを選択します。

e. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

f. クラスターの記述名を入力します。



重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの 予約されたリソース名のエラーを解決する を参照してください。

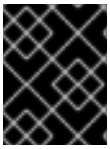
g. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの ~/**.azure**/ ディレクトリーに保存します。これにより、インストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作成するときにプロファイルをロードできるようになります。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

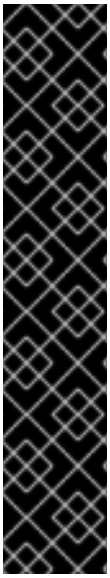


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

7.4.6. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.4.7. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

7.4.8. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または

OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.4.9. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

7.5. カスタマイズによる AZURE へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが Microsoft Azure 上にプロビジョニングするインフラストラクチャーに、カスタマイズしたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

7.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- 暗号化のために Azure 環境を準備 した (顧客管理の暗号化キーを使用する場合)。

7.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.5.3. クラスターノードの SSH アクセス用のキーペアの生成

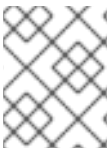
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.5.4. Azure Marketplace オファリングの使用

Azure Marketplace を使用すると、OpenShift Container Platform クラスターをデプロイできます。これは、Azure を通じて従量課金制 (時間単位、コア単位) で請求され、Red Hat の直接サポートも受けることができます。

Azure Marketplace オファリングを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に Azure Marketplace イメージを取得する必要があります。インストールプログラムは、このイメージを使用してワーカーまたはコントロールプレーンノードをデプロイします。イメージを取得するときは、次の点を考慮してください。

- イメージは同じですが、Azure Marketplace のパブリシャーは地域によって異なります。北米にお住まいの場合は、**redhat** をパブリッシャーとして指定してください。EMEA にお住まいの場合は、**redhat-limited** をパブリッシャーとして指定してください。
- このオファーには、**rh-ocp-worker** SKU と **rh-ocp-worker-gen1** SKU が含まれています。**rh-ocp-worker** SKU は、Hyper-V 世代のバージョン 2 VM イメージを表します。OpenShift Container Platform で使用されるデフォルトのインスタンスタイプは、バージョン 2 と互換性があります。バージョン 1 のみと互換性のあるインスタンスタイプを使用する場合は、**rh-ocp-worker-gen1** SKU に関連付けられたイメージを使用します。**rh-ocp-worker-gen1** SKU は、Hyper-V バージョン 1 VM イメージを表します。

重要

Azure マーケットプレイスを使用したイメージのインストールは、64 ビット ARM インスタンスを備えたクラスターではサポートされていません。

前提条件

- Azure CLI クライアント **(az)** をインストールしている。
- お客様の Azure アカウントにはオファーのエンタイトルメントがあり、Azure CLI クライアントを使用してこのアカウントにログインしている。

手順

1. 以下のいずれかのコマンドを実行して、利用可能なすべての OpenShift Container Platform イメージを表示します。
   - 北米:

     ```
     $ az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
     ```

     出力例

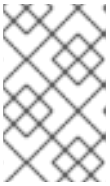
     ```
     Offer          Publisher      Sku                Urn                                                                  Version
     -------------  -------------  -----------------  -------------------------------------------------------------------  ----  -----------------
     rh-ocp-worker  RedHat         rh-ocp-worker      RedHat:rh-ocp-worker:rh-ocp-worker:413.92.2023101700                 413.92.2023101700
     rh-ocp-worker  RedHat         rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-gen1:413.92.2023101700        413.92.2023101700
     ```

   - EMEA:

     ```
     $ az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
     ```

出力例

```
Offer          Publisher       Sku                 Urn
Version
-------------  --------------  ------------------  ------------------------------------------------------------
----           -----------------
rh-ocp-worker  redhat-limited  rh-ocp-worker       redhat-limited:rh-ocp-worker:rh-ocp-
worker:413.92.2023101700            413.92.2023101700
rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
worker-gen1:413.92.2023101700       413.92.2023101700
```



注記

コンピュートおよびコントロールプレーンノードで利用可能な最新のイメージを使用します。必要に応じて、VM はインストールプロセスの一部として自動的にアップグレードされます。

2. 次のいずれかのコマンドを実行して、オファーのイメージを調べます。
   - 北米:

     ```
     $ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - EMEA:

     ```
     $ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

3. 次のコマンドのいずれかを実行して、オファーの条件を確認します。
   - 北米:

     ```
     $ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - EMEA:

     ```
     $ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

4. 次のコマンドのいずれかを実行して、オファリングの条件に同意します。
   - 北米:

     ```
     $ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - EMEA:

     ```
     $ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

5. オファーのイメージの詳細を記録します。クラスターをデプロイする前に、**install-config.yaml** ファイルの **compute** セクションを、**publisher**、**offer**、**sku**、および **version** の値で更新する必要があります。**controlPlane** セクションを更新して、指定されたイメージの詳細でコントロールプレーンマシンをデプロイするか、**defaultMachinePlatform** セクションを更

新して、指定されたイメージの詳細を使用してコントロールプレーンとコンピュートマシンの両方をデプロイすることもできます。コントロールプレーンおよびコンピュートノード用に利用可能な最新のイメージを使用します。

Azure Marketplace コンピュートノードを含む install-config.yaml ファイルのサンプル

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    azure:
      type: Standard_D4s_v5
      osImage:
        publisher: redhat
        offer: rh-ocp-worker
        sku: rh-ocp-worker
        version: 413.92.2023101700
  replicas: 3
```

7.5.5. インストールプログラムの取得

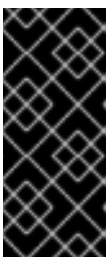
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.5.6. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルを使用してクラスターをインストールしている場合は、そのアプリケーション ID とパスワードが必要です。
- システムが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、インストールプログラムを実行する仮想マシン上でそれが有効になっています。
- ユーザーが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、次の前提条件を満たしている必要があります。
  - そのクライアント ID がある。
  - これは、インストールプログラムを実行する仮想マシンに割り当てられている。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルまたはマネージド ID を使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。

2. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。
インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。

iii. サブスクリプションの次の Azure パラメーター値を入力します。

- **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
- **azure tenant id** テナント ID を入力します。

iv. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアント ID** の入力を求められたら、次のいずれかを行います。

- サービスプリンシパルを使用している場合は、そのアプリケーション ID を入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、そのクライアント ID を指定します。

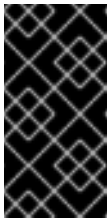
v. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアントシークレット** の入力を求められたら、次のいずれかを実行します。

- サービスプリンシパルを使用している場合は、そのパスワードを入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

vi. クラスターをデプロイするリージョンを選択します。

vii. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

viii. クラスターの記述名を入力します。



重要

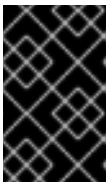
パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

3. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

注記

3 ノードクラスターをインストールする場合は、必ず **compute.replicas** パラメーターを **0** に設定してください。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細については、「Azure に 3 ノードクラスターをインストールする」を参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの **~/.azure/** ディレクトリーに保存します。これにより、インストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作成するときにプロファイルをロードできるようになります。

関連情報

- Azure のインストール設定パラメーター

7.5.6.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

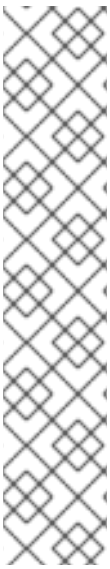
表7.1 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

重要

premiumIO パラメーターが **true** に設定されている Azure 仮想マシンを使用する必要があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

7.5.6.2. Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure インスタンスタイプは OpenShift Container Platform でテストされています。

例7.24 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

7.5.6.3. 64 ビット ARM インフラストラクチャー上の Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure Azure64 インスタンスタイプは OpenShift Container Platform でテストされています。

例7.25 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

7.5.6.4. Azure VM の信頼された起動の有効化

Azure にクラスターをインストールするときに、セキュアブート と 仮想化された信頼できるプラットフォームモジュール という 2 つの信頼された起動機能を有効にできます。

これらの機能をサポートする 仮想マシンのサイズ については、仮想マシンのサイズに関する Azure のドキュメントを参照してください。

重要

信頼できる起動はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

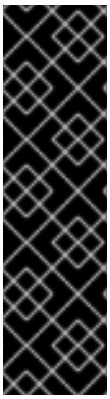
- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: TrustedLaunch 2
        trustedLaunch:
          uefiSettings:
            secureBoot: Enabled 3
            virtualizedTrustedPlatformModule: Enabled 4
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定すると、それぞれコントロールプレーンまたはコンピュートノードのみで信頼された起動が有効になります。すべてのノードで信頼できる起動を有効にするには、**platform.azure.defaultMachinePlatform** を指定します。
2. 信頼できる起動機能を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。

7.5.6.5. Confidential VM の有効化

クラスターをインストールするときに、Confidential 仮想マシンを有効にできます。コンピューティングノード、コンピュートノード、またはすべてのノードに対して Confidential 仮想マシンを有効にできます。



重要

Confidential VMs の使用はテクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

次の仮想マシンサイズの Confidential VM を使用できます。

- DCasv5 シリーズ
- DCadsv5 シリーズ
- ECasv5 シリーズ
- ECadsv5 シリーズ

重要

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: ConfidentialVM 2
        confidentialVM:
          uefiSettings:
            secureBoot: Enabled 3
            virtualizedTrustedPlatformModule: Enabled 4
      osDisk:
        securityProfile:
          securityEncryptionType: VMGuestStateOnly 5
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定して、それぞれコントロールプレーンまたはコンピュートノードのみに Confidential 仮想マシンをデプロイし
2. Confidential VM を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。
5. 仮想マシンゲストの状態を暗号化するには、**VMGuestStateOnly** を指定します。

7.5.6.6. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      encryptionAtHost: true
      ultraSSDCapability: Enabled
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      ultraSSDCapability: Enabled
      type: Standard_D2s_v3
```

```
      encryptionAtHost: true
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 11
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    defaultMachinePlatform:
      osImage: 12
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      ultraSSDCapability: Enabled
    baseDomainResourceGroupName: resource_group 13
    region: centralus 14
    resourceGroupName: existing_resource_group 15
    outboundType: Loadbalancer
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 16
fips: false 17
sshKey: ssh-ed25519 AAAA... 18
```

1 10 14 16 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1

つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンのリストを指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 オプション: コントロールプレーンとコンピュートマシンを起動するために使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) イメージ。**platform.azure.defaultMachinePlatform.osImage** の下の **publisher**、**offer**、**sku**、および **version** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**controlPlane.platform.azure.osImage** または **compute.platform.azure.osImage** の下のパラメーターが設定されている場合、それらは **platform.azure.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

13 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

15 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

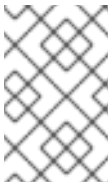
17 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

18 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

7.5.6.7. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.**

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

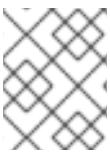
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

関連情報

- 高速ネットワークの詳細については、Accelerated Networking for Microsoft Azure VMs を参照してください。

7.5.7. Azure のユーザー定義タグを設定する

OpenShift Container Platform では、タグを使用して、リソースをグループ化し、リソースへのアクセスとコストを管理できます。**install-config.yaml** ファイルで Azure リソースのタグを定義できるのは、OpenShift Container Platform クラスターの作成時のみです。クラスターの作成後は、ユーザー定義タグを変更することはできません。

ユーザー定義タグのサポートは、Azure Public Cloud で作成されたリソースでのみ利用できます。ユーザー定義タグは、OpenShift Container Platform 4.16 にアップグレードされた OpenShift Container Platform クラスターではサポートされていません。

ユーザー定義および OpenShift Container Platform 固有タグは、OpenShift Container Platform インストーラーとそのコアオペレーター (マシン API プロバイダーの Azure Operator、Cluster Ingress Operator、Cluster Image Registry Operator など) によって作成されたリソースのみに適用されます。

デフォルトでは、OpenShift Container Platform インストーラーは、OpenShift Container Platform タグを Azure リソースにアタッチします。ユーザーは、これらの OpenShift Container Platform タグにアクセスできません。

次の **install-config.yaml** ファイルに示すように、**install-config.yaml** ファイルの **.platform.azure.userTags** フィールドを使用して、ユーザー定義タグのリストを定義できます。

install-config.yaml ファイルのサンプル

```
additionalTrustBundlePolicy: Proxyonly (1)
apiVersion: v1
baseDomain: catchall.azure.devcluster.openshift.com (2)
compute: (3)
- architecture: amd64
  hyperthreading: Enabled (4)
  name: worker
  platform: {}
  replicas: 3
controlPlane: (5)
  architecture: amd64
  hyperthreading: Enabled (6)
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: user (7)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (8)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: os4-common (9)
    cloudName: AzurePublicCloud (10)
    outboundType: Loadbalancer
    region: southindia (11)
    userTags: (12)
      createdBy: user
      environment: dev
```

1. 信頼バンドルポリシーを定義します。
2. 必須。**baseDomain** パラメーターは、クラウドプロバイダーのベースドメインを指定します。インストールプログラムはこの値の入力を求めるプロンプトを出します。
3. コンピューティングを設定するマシンの設定。**compute** セクションはマッピングのシーケンスです。異なるデータ構造の要件を満たすには、**compute** セクションの最初の行がハイフン **-** で始まる必要があります。これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。
4. 同時マルチスレッドまたは **hyperthreading** を有効または無効にします。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。
5. コントロールプレーンを設定するマシンの設定。**controlPlane** セクションは単一のマッピングです。**controlPlane** セクションの最初の行は、ハイフン **-** を使用しないでください。1 つのコントロールプレーンプールのみを使用できます。これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。
6. 同時マルチスレッドまたは **hyperthreading** を有効または無効にします。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。
7. インストールプログラムはこの値の入力を求めるプロンプトを出します。
8. インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。
9. Azure DNS ゾーンのベースドメインのリソースグループを指定します。
10. Azure クラウド環境の名前を指定します。**cloudName** フィールドを使用して、Azure API エンドポイントで Azure SDK を設定できます。値を指定しない場合、デフォルト値は、Azure Public Cloud です。
11. 必須。クラスターをホストする Azure リージョンの名前を指定します。インストールプログラムはこの値の入力を求めるプロンプトを出します。
12. インストールプログラムが作成するすべての Azure リソースにタグとして追加する追加のキーと値を定義します。

ユーザー定義タグには、次の制限があります。

- タグキーは最大 128 文字です。
- タグキーは文字で始まり、文字、数字、またはアンダースコアで終わる必要があり、文字、数字、アンダースコア、ピリオド、およびハイフンのみを含めることができます。
- タグキーは大文字と小文字を区別しません。
- タグキーを **name** にすることはできません。**kubernetes.io**、**openshift.io**、**microsoft**、**azure**、および **windows** などの接頭辞を付けることはできません。
- タグ値は最大 256 文字です。

- リソースグループとリソースに最大 10 個のタグを設定できます。

Azure タグの詳細については、Azure のユーザー定義タグ を参照してください。

7.5.8. Azure のユーザー定義タグのクエリー

OpenShift Container Platform クラスターを作成したら、Azure リソース用に定義されたタグのリストにアクセスできます。OpenShift Container Platform タグの形式は、**kubernetes.io_cluster.<cluster_id>:owned** です。**cluster_id** パラメーターは、**config.openshift.io/Infrastructure** にある **.status.infrastructureName** の値です。

- 次のコマンドを実行して、Azure リソース用に定義されたタグをクエリーします。

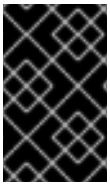
```
$ oc get infrastructures.config.openshift.io cluster -o=jsonpath-as-
json='{.status.platformStatus.azure.resourceTags}'
```

出力例

```
[
  [
    {
      "key": "createdBy",
      "value": "user"
    },
    {
      "key": "environment",
      "value": "dev"
    }
  ]
]
```

7.5.9. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

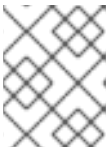
```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。

5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.5.10. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。

- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように Azure クラスターを設定する の手順に従ってください。

7.5.10.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
    ...
```

5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```

重要

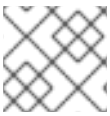
手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

7.5.10.2. 短期認証情報を使用するように Azure クラスターを設定する

Microsoft Entra Workload ID を使用するクラスターをインストールするには、Cloud Credential Operator ユーティリティーを設定し、クラスターに必要な Azure リソースを作成する必要があります。

7.5.10.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- 次の権限で使用する **ccoctl** ユーティリティー用のグローバル Microsoft Azure アカウントが作成されました。

  例7.26 必要な Azure 権限

  - Microsoft.Resources/subscriptions/resourceGroups/read
  - Microsoft.Resources/subscriptions/resourceGroups/write
  - Microsoft.Resources/subscriptions/resourceGroups/delete
  - Microsoft.Authorization/roleAssignments/read
  - Microsoft.Authorization/roleAssignments/delete
  - Microsoft.Authorization/roleAssignments/write
  - Microsoft.Authorization/roleDefinitions/read
  - Microsoft.Authorization/roleDefinitions/write
  - Microsoft.Authorization/roleDefinitions/delete
  - Microsoft.Storage/storageAccounts/listkeys/action
  - Microsoft.Storage/storageAccounts/delete
  - Microsoft.Storage/storageAccounts/read
  - Microsoft.Storage/storageAccounts/write
  - Microsoft.Storage/storageAccounts/blobServices/containers/write
  - Microsoft.Storage/storageAccounts/blobServices/containers/delete
  - Microsoft.Storage/storageAccounts/blobServices/containers/read
  - Microsoft.ManagedIdentity/userAssignedIdentities/delete
  - Microsoft.ManagedIdentity/userAssignedIdentities/read
  - Microsoft.ManagedIdentity/userAssignedIdentities/write
  - Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/read
  - Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/write
  - Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/delete
  - Microsoft.Storage/register/action
  - Microsoft.ManagedIdentity/register/action

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
```

```
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

7.5.10.2.2. Cloud Credential Operator ユーティリティーを使用した Azure リソースの作成

ccoctl azure create-all コマンドを使用して、Azure リソースの作成を自動化できます。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。
- Azure CLI を使用して Microsoft Azure アカウントにアクセスします。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

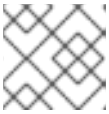
```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3.

CredentialsRequest オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。



注記

このコマンドの実行には少し時間がかかる場合があります。

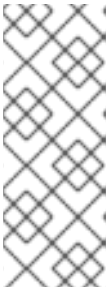
3. **ccoctl** ユーティリティーが Azure 認証情報を自動的に検出できるようにするには、次のコマンドを実行して Azure CLI にログインします。

```
$ az login
```

4. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl azure create-all \
  --name=<azure_infra_name> \ 1
  --output-dir=<ccoctl_output_dir> \ 2
  --region=<azure_region> \ 3
  --subscription-id=<azure_subscription_id> \ 4
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 5
  --dnszone-resource-group-name=<azure_dns_zone_resource_group_name> \ 6
  --tenant-id=<azure_tenant_id> 7
```

1. トラッキングに使用される、作成されたすべての Azure リソースのユーザー定義名を指定します。
2. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
3. クラウドリソースが作成される Azure リージョンです。
4. 使用する Azure サブスクリプション ID を指定します。
5. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
6. クラスターのベースドメイン Azure DNS ゾーンを含むリソースグループの名前を指定します。
7. 使用する Azure テナント ID を指定します。



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

追加のオプションパラメーターとその使用方法の説明を表示するには、**azure create-all --help** コマンドを実行します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
azure-ad-pod-identity-webhook-config.yaml
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-azure-cloud-credentials-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capz-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-disk-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-file-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-azure-cloud-credentials-credentials.yaml
```

Azure をクエリーすることで、Microsoft Entra ID サービスアカウントが作成されていることを確認できます。詳細は、Entra ID サービスアカウントのリストに関する Azure ドキュメントを参照してください。

7.5.10.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

2. 既存のリソースグループを使用する代わりに、**ccoctl** ユーティリティーを使用して新しい Azure リソースグループを作成した場合は、次のように **install-config.yaml** の **resourceGroupName** パラメーターを変更します。

設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
# ...
platform:
  azure:
    resourceGroupName: <azure_infra_name> 1
# ...
```

1 この値は、**ccoctl azure create-all** コマンドの **--name** 引数で指定された Azure リソースのユーザー定義名と一致する必要があります。

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

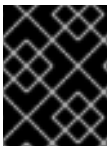
```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

7.5.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

7.5.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

7.5.13. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.5.14. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

7.6. ネットワークのカスタマイズによる AZURE へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが Microsoft Azure 上にプロビジョニングするインフラストラクチャーに、カスタマイズしたネットワーク設定でクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

7.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- 暗号化のために Azure 環境を準備 した (顧客管理の暗号化キーを使用する場合)。

7.6.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.6.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

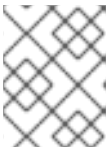
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

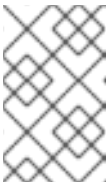
一部のディストリビューションでは、**~/.ssh/id_rsa** および **~/.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.6.4. インストールプログラムの取得

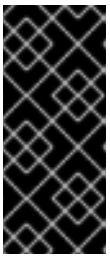
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.6.5. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルを使用してクラスターをインストールしている場合は、そのアプリケーション ID とパスワードが必要です。
- システムが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、インストールプログラムを実行する仮想マシン上でそれが有効になっています。

- ユーザーが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、次の前提条件を満たしている必要があります。
  - そのクライアント ID がある。
  - これは、インストールプログラムを実行する仮想マシンに割り当てられている。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルまたはマネージド ID を使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
   このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。

2. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **azure** を選択します。
      インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。

      iii. サブスクリプションの次の Azure パラメーター値を入力します。

- **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
- **azure tenant id** テナント ID を入力します。

iv. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアント ID** の入力を求められたら、次のいずれかを行います。

- サービスプリンシパルを使用している場合は、そのアプリケーション ID を入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、そのクライアント ID を指定します。

v. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアントシークレット** の入力を求められたら、次のいずれかを実行します。

- サービスプリンシパルを使用している場合は、そのパスワードを入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

vi. クラスターをデプロイするリージョンを選択します。

vii. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

viii. クラスターの記述名を入力します。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

3. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。
4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの **~/.azure/** ディレクトリーに保存します。これにより、イン

ストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作成するときにプロファイルをロードできるようになります。

関連情報

- Azure のインストール設定パラメーター

7.6.5.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

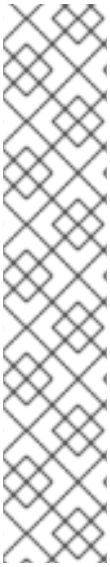
表7.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

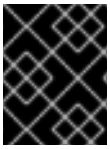


注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。



重要

premiumIO パラメーターが **true** に設定されている Azure 仮想マシンを使用する必要があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

7.6.5.2. Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure インスタンスタイプは OpenShift Container Platform でテストされています。

例7.27 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***

- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

7.6.5.3. 64 ビット ARM インフラストラクチャー上の Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure Azure64 インスタンスタイプは OpenShift Container Platform でテストされています。

例7.28 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

7.6.5.4. Azure VM の信頼された起動の有効化

Azure にクラスターをインストールするときに、セキュアブート と 仮想化された信頼できるプラットフォームモジュール という 2 つの信頼された起動機能を有効にできます。

これらの機能をサポートする 仮想マシンのサイズ については、仮想マシンのサイズに関する Azure のドキュメントを参照してください。

重要

信頼できる起動はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
```

```
    securityType: TrustedLaunch 2
    trustedLaunch:
      uefiSettings:
        secureBoot: Enabled 3
        virtualizedTrustedPlatformModule: Enabled 4
```

1 **controlPlane.platform.azure** または **compute.platform.azure** を指定すると、それぞれコントロールプレーンまたはコンピュートノードのみで信頼された起動が有効になります。すべてのノードで信頼できる起動を有効にするには、**platform.azure.defaultMachinePlatform** を指定します。

2 信頼できる起動機能を有効にします。

3 Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。

4 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。

7.6.5.5. Confidential VM の有効化

クラスターをインストールするときに、Confidential 仮想マシンを有効にできます。コンピューティングノード、コンピュートノード、またはすべてのノードに対して Confidential 仮想マシンを有効にできます。

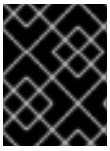
重要

Confidential VMs の使用はテクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

次の仮想マシンサイズの Confidential VM を使用できます。

- DCasv5 シリーズ
- DCadsv5 シリーズ
- ECasv5 シリーズ
- ECadsv5 シリーズ



重要

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

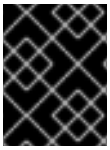
- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: ConfidentialVM 2
        confidentialVM:
          uefiSettings:
            secureBoot: Enabled 3
            virtualizedTrustedPlatformModule: Enabled 4
      osDisk:
        securityProfile:
          securityEncryptionType: VMGuestStateOnly 5
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定して、それぞれコントロールプレーンまたはコンピュートノードのみに Confidential 仮想マシンをデプロイします。すべてのノードに Confidential 仮想マシンをデプロイするには、**platform.azure.defaultMachinePlatform** を指定します。
2. Confidential VM を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。
5. 仮想マシンゲストの状態を暗号化するには、**VMGuestStateOnly** を指定します。

7.6.5.6. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    azure:
```

```
      encryptionAtHost: true
      ultraSSDCapability: Enabled
      osDisk:
        diskSizeGB: 1024 5
        diskType: Premium_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      type: Standard_D8s_v3
  replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      ultraSSDCapability: Enabled
      type: Standard_D2s_v3
      encryptionAtHost: true
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking: 11
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 12
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    defaultMachinePlatform:
```

```
      osImage: 13
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      ultraSSDCapability: Enabled
    baseDomainResourceGroupName: resource_group 14
    region: centralus 15
    resourceGroupName: existing_resource_group 16
    outboundType: Loadbalancer
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 17
fips: false 18
sshKey: ssh-ed25519 AAAA... 19
```

1 10 15 17 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 11 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンのリストを指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

12 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

13 オプション: コントロールプレーンとコンピュートマシンを起動するために使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) イメージ。**platform.azure.defaultMachinePlatform.osImage** の下の **publisher**、**offer**、**sku**、および **version** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**controlPlane.platform.azure.osImage** または **compute.platform.azure.osImage** の下のパラメーターが設定されている場合、それらは **platform.azure.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

14 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

16 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

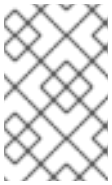
18 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

19 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

7.6.5.7. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

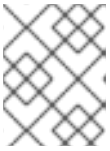
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

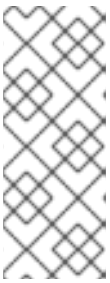
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

7.6.6. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**

これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。

注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

重要

CIDR 範囲 **172.17.0.0/16** は libVirt によって予約されています。この範囲、またはこの範囲と重複する範囲をクラスター内のネットワークに使用することはできません。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 でネットワークプラグインをさらにカスタマイズできます。

7.6.7. 高度なネットワーク設定の指定

ネットワークプラグインに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 次の例のように、**cluster-network-03-config.yml** ファイルでクラスターの高度なネットワーク設定を指定します。

OVN-Kubernetes ネットワークプロバイダーを有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

```
defaultNetwork:
  ovnKubernetesConfig:
    ipsecConfig:
      mode: Full
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

7.6.8. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

7.6.8.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表7.3 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 `spec:` `  clusterNetwork:` `  - cidr: 10.128.0.0/19` `  hostPrefix: 23` `  - cidr: 10.128.32.0/19` `  hostPrefix: 23`

フィールド	型	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表7.4 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表7.5 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表7.6 ovnKubernetesConfig.ipv4 object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表7.7 ovnKubernetesConfig.ipv6 object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表7.8 policyAuditConfig オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表7.9 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

フィールド	型	説明
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表7.10 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表7.11 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表7.12 **ipsecConfig** オブジェクト

フィールド	型	説明

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

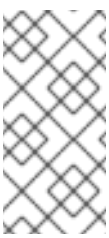
```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表7.13 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	型	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

7.6.9. OVN-Kubernetes を使用したハイブリッドネットワークの設定

OVN-Kubernetes ネットワークプラグインを使用してハイブリッドネットワークを使用するようにクラスターを設定できます。これにより、異なるノードのネットワーク設定をサポートするハイブリッドクラスターが可能になります。

前提条件

- **install-config.yaml** ファイルで **networking.networkType** パラメーターの **OVNKubernetes** を定義していること。詳細は、選択したクラウドプロバイダーでの OpenShift Container Platform ネットワークのカスタマイズの設定についてのインストールドキュメントを参照してください。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

ここでは、以下のようになります。

<installation_directory>
クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
EOF
```

ここでは、以下のようになります。

<installation_directory>
クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

3. **cluster-network-03-config.yml** ファイルをエディターで開き、以下の例のようにハイブリッドネットワークで OVN-Kubernetes を設定します。

ハイブリッドネットワーク設定の指定

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork: 1
        - cidr: 10.132.0.0/14
          hostPrefix: 23
```

1 追加のオーバーレイネットワーク上のノードに使用される CIDR 設定を指定します。**hybridClusterNetwork** CIDR は **clusterNetwork** CIDR と重複できません。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

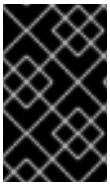
5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

関連情報

- 高速ネットワークの詳細については、Accelerated Networking for Microsoft Azure VMs を参照してください。

7.6.10. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。

2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

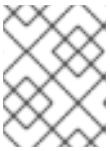
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.6.11. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように Azure クラスターを設定する の手順に従ってください。

7.6.11.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
```

```
credentialsMode: Manual
# ...
```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
  ...
```

5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```

重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

7.6.11.2. 短期認証情報を使用するように Azure クラスターを設定する

Microsoft Entra Workload ID を使用するクラスターをインストールするには、Cloud Credential Operator ユーティリティーを設定し、クラスターに必要な Azure リソースを作成する必要があります。

7.6.11.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- 次の権限で使用する **ccoctl** ユーティリティー用のグローバル Microsoft Azure アカウントが作成されました。

  例7.29 必要な Azure 権限

  - Microsoft.Resources/subscriptions/resourceGroups/read
  - Microsoft.Resources/subscriptions/resourceGroups/write
  - Microsoft.Resources/subscriptions/resourceGroups/delete
  - Microsoft.Authorization/roleAssignments/read
  - Microsoft.Authorization/roleAssignments/delete
  - Microsoft.Authorization/roleAssignments/write
  - Microsoft.Authorization/roleDefinitions/read
  - Microsoft.Authorization/roleDefinitions/write
  - Microsoft.Authorization/roleDefinitions/delete
  - Microsoft.Storage/storageAccounts/listkeys/action
  - Microsoft.Storage/storageAccounts/delete
  - Microsoft.Storage/storageAccounts/read
  - Microsoft.Storage/storageAccounts/write
  - Microsoft.Storage/storageAccounts/blobServices/containers/write
  - Microsoft.Storage/storageAccounts/blobServices/containers/delete
  - Microsoft.Storage/storageAccounts/blobServices/containers/read
  - Microsoft.ManagedIdentity/userAssignedIdentities/delete
  - Microsoft.ManagedIdentity/userAssignedIdentities/read
  - Microsoft.ManagedIdentity/userAssignedIdentities/write
  - Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/read

- Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/write
- Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/delete
- Microsoft.Storage/register/action
- Microsoft.ManagedIdentity/register/action

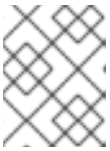
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

7.6.11.2.2. Cloud Credential Operator ユーティリティーを使用した Azure リソースの作成

ccoctl azure create-all コマンドを使用して、Azure リソースの作成を自動化できます。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。
- Azure CLI を使用して Microsoft Azure アカウントにアクセスします。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
```

```
--included \ 1
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
--to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. **ccoctl** ユーティリティーが Azure 認証情報を自動的に検出できるようにするには、次のコマンドを実行して Azure CLI にログインします。

```
$ az login
```

4. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl azure create-all \
  --name=<azure_infra_name> \ 1
  --output-dir=<ccoctl_output_dir> \ 2
  --region=<azure_region> \ 3
  --subscription-id=<azure_subscription_id> \ 4
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 5
  --dnszone-resource-group-name=<azure_dns_zone_resource_group_name> \ 6
  --tenant-id=<azure_tenant_id> 7
```

1. トラッキングに使用される、作成されたすべての Azure リソースのユーザー定義名を指定します。
2. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
3. クラウドリソースが作成される Azure リージョンです。
4. 使用する Azure サブスクリプション ID を指定します。
5. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
6. クラスターのベースドメイン Azure DNS ゾーンを含むリソースグループの名前を指定します。
7. 使用する Azure テナント ID を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

追加のオプションパラメーターとその使用方法の説明を表示するには、**azure create-all --help** コマンドを実行します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
azure-ad-pod-identity-webhook-config.yaml
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-azure-cloud-credentials-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capz-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-disk-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-file-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-azure-cloud-credentials-credentials.yaml
```

Azure をクエリーすることで、Microsoft Entra ID サービスアカウントが作成されていることを確認できます。詳細は、Entra ID サービスアカウントのリストに関する Azure ドキュメントを参照してください。

7.6.11.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

設定ファイルのサンプルスニペット

```
-
```

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

2. 既存のリソースグループを使用する代わりに、**ccoctl** ユーティリティーを使用して新しい Azure リソースグループを作成した場合は、次のように **install-config.yaml** の **resourceGroupName** パラメーターを変更します。

   設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
# ...
platform:
  azure:
    resourceGroupName: <azure_infra_name> 1
# ...
```

   1 この値は、**ccoctl azure create-all** コマンドの **--name** 引数で指定された Azure リソースのユーザー定義名と一致する必要があります。

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

7.6.12. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

7.6.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

7.6.14. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.6.15. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

7.7. AZURE のクラスターの既存 VNET へのインストール

OpenShift Container Platform バージョン 4.16 では、Microsoft Azure 上の既存の Azure Virtual Network (VNet) にクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

7.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- 暗号化のために Azure 環境を準備 した (顧客管理の暗号化キーを使用する場合)。

7.7.2. OpenShift Container Platform クラスターでの VNet の再利用について

OpenShift Container Platform 4.16 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

7.7.2.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加のネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。

注記

デフォルトでは、**install-config.yaml** ファイルでアベイラビリティゾーンを指定すると、インストールプログラムはコントロールプレーンマシンとコンピューティングマシンを リージョン 内の これらのアベイラビリティゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

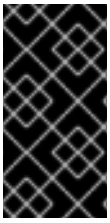
- 指定されたサブネットがすべて存在します。
- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットがあります
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。必要な場合に、インストールプログラムはコントロールプレーンおよびワーカーノードを管理するパブリックロードバランサーを作成し、Azure はパブリック IP アドレスをそれらに割り当てます。

注記

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

7.7.2.1.1. ネットワークセキュリティーグループの要件

コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。



重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表7.14 必須ポート

ポート	説明	コントロールプレーン	Compute
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	
22623	マシンをプロビジョニングするためのマシン設定サーバーへの内部通信を許可します。	x	

1. Azure Firewall を使用してインターネットアクセスを制限している場合は、Azure API を許可するように Azure Firewall を設定できます。ネットワークセキュリティーグループルールは必要ありません。

重要

現在、マシン設定サーバーエンドポイントをブロックまたは制限する方法はサポートされていません。マシン設定サーバーは、既存の設定または状態を持たない新しくプロビジョニングされたマシンが設定を取得できるように、ネットワークに公開する必要があります。このモデルでは、信頼のルートは証明書署名要求 (CSR) エンドポイントであり、kubelet がクラスターに参加するための承認のために証明書署名要求を送信する場所です。このため、シークレットや証明書などの機密情報を配布するためにマシン設定を使用しないでください。

マシン設定サーバーエンドポイント、ポート 22623 および 22624 がベアメタルシナリオで確実に保護されるようにするには、顧客は適切なネットワークポリシーを設定する必要があります。

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

関連情報

- OpenShift SDN ネットワークプラグインについて
- ファイアウォールの設定

7.7.2.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

7.7.2.3. クラスター間の分離

クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

7.7.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.7.4. クラスターノードの SSH アクセス用のキーペアの生成

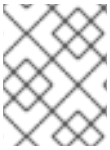
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> (1)
```

(1) 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

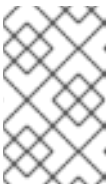
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.7.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

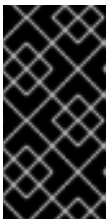
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.7.6. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルを使用してクラスターをインストールしている場合は、そのアプリケーション ID とパスワードが必要です。
- システムが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、インストールプログラムを実行する仮想マシン上でそれが有効になっています。
- ユーザーが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、次の前提条件を満たしている必要があります。
  - そのクライアント ID がある。
  - これは、インストールプログラムを実行する仮想マシンに割り当てられている。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルまたはマネージド ID を使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
   このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。
2. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

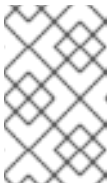
   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。
インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。

iii. サブスクリプションの次の Azure パラメーター値を入力します。

- **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
- **azure tenant id** テナント ID を入力します。

iv. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアント ID** の入力を求められたら、次のいずれかを行います。

- サービスプリンシパルを使用している場合は、そのアプリケーション ID を入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、そのクライアント ID を指定します。

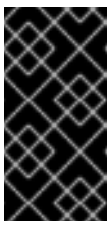
v. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアントシークレット** の入力を求められたら、次のいずれかを実行します。

- サービスプリンシパルを使用している場合は、そのパスワードを入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

vi. クラスターをデプロイするリージョンを選択します。

vii. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

viii. クラスターの記述名を入力します。

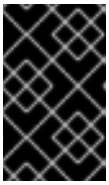


重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

3. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの **~/.azure/** ディレクトリーに保存します。これにより、インストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作成するときにプロファイルをロードできるようになります。

関連情報

- Azure のインストール設定パラメーター

7.7.6.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表7.15 最小リソース要件

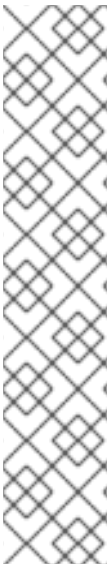
マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、そ

の他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

重要

premiumIO パラメーターが **true** に設定されている Azure 仮想マシンを使用する必要があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

7.7.6.2. Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure インスタンスタイプは OpenShift Container Platform でテストされています。

例7.30 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***

- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

7.7.6.3. 64 ビット ARM インフラストラクチャー上の Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure Azure64 インスタンスタイプは OpenShift Container Platform でテストされています。

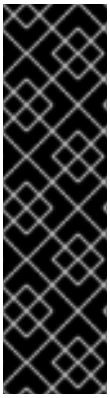
例7.31 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

7.7.6.4. Azure VM の信頼された起動の有効化

Azure にクラスターをインストールするときに、セキュアブート と 仮想化された信頼できるプラットフォームモジュール という 2 つの信頼された起動機能を有効にできます。

これらの機能をサポートする 仮想マシンのサイズ については、仮想マシンのサイズに関する Azure のドキュメントを参照してください。



重要

信頼できる起動はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
 platform:
  azure:
   settings:
    securityType: TrustedLaunch 2
    trustedLaunch:
     uefiSettings:
      secureBoot: Enabled 3
      virtualizedTrustedPlatformModule: Enabled 4
```

1 **controlPlane.platform.azure** または **compute.platform.azure** を指定すると、それぞれコントロールプレーンまたはコンピュートノードのみで信頼された起動が有効になります。すべてのノードで信頼できる起動を有効にするには、**platform.azure.defaultMachinePlatform** を指定します。

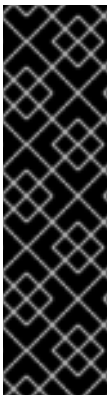
2 信頼できる起動機能を有効にします。

3 Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。

4 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。

7.7.6.5. Confidential VM の有効化

クラスターをインストールするときに、Confidential 仮想マシンを有効にできます。コンピューティングノード、コンピュートノード、またはすべてのノードに対して Confidential 仮想マシンを有効にできます。



重要

Confidential VMs の使用はテクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

次の仮想マシンサイズの Confidential VM を使用できます。

- DCasv5 シリーズ
- DCadsv5 シリーズ
- ECasv5 シリーズ
- ECadsv5 シリーズ

重要

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

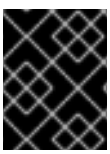
- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: ConfidentialVM 2
        confidentialVM:
          uefiSettings:
            secureBoot: Enabled 3
            virtualizedTrustedPlatformModule: Enabled 4
      osDisk:
        securityProfile:
          securityEncryptionType: VMGuestStateOnly 5
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定して、それぞれコントロールプレーンまたはコンピュートノードのみに Confidential 仮想マシンをデプロイします。すべてのノードに Confidential 仮想マシンをデプロイするには、**platform.azure.defaultMachinePlatform** を指定します。
2. Confidential VM を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。
5. 仮想マシンゲストの状態を暗号化するには、**VMGuestStateOnly** を指定します。

7.7.6.6. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
```

```
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      encryptionAtHost: true
      ultraSSDCapability: Enabled
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      ultraSSDCapability: Enabled
      type: Standard_D2s_v3
      encryptionAtHost: true
      osDisk:
        diskSizeGB: 512 (8)
        diskType: Standard_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
```

```
  networkType: OVNKubernetes (11)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    defaultMachinePlatform:
      osImage: (12)
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      ultraSSDCapability: Enabled
    baseDomainResourceGroupName: resource_group (13)
    region: centralus (14)
    resourceGroupName: existing_resource_group (15)
    networkResourceGroupName: vnet_resource_group (16)
    virtualNetwork: vnet (17)
    controlPlaneSubnet: control_plane_subnet (18)
    computeSubnet: compute_subnet (19)
    outboundType: Loadbalancer
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' (20)
fips: false (21)
sshKey: ssh-ed25519 AAAA... (22)
```

(1) (10) (14) (20) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(2) (6) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (7) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(4) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

(5) (8) 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

(9) マシンをデプロイするゾーンのリストを指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

(11) インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

OVNKubernetes のみです。

12 オプション: コントロールプレーンとコンピュートマシンを起動するために使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) イメージ。**platform.azure.defaultMachinePlatform.osImage** の下の **publisher**、**offer**、**sku**、および **version** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**controlPlane.platform.azure.osImage** または **compute.platform.azure.osImage** の下のパラメーターが設定されている場合、それらは **platform.azure.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

13 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

15 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

16 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

17 既存の VNet を使用する場合は、その名前を指定します。

18 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

19 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

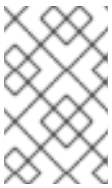
21 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

22 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

7.7.6.7. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする

trusted-ca-bundle 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

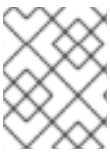
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

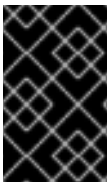
cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

関連情報

- 高速ネットワークの詳細については、Accelerated Networking for Microsoft Azure VMs を参照してください。

7.7.7. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

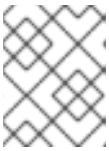
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.7.8. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように Azure クラスターを設定する の手順に従ってください。

7.7.8.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

   サンプル CredentialsRequest オブジェクト

   ```
   apiVersion: cloudcredential.openshift.io/v1
   kind: CredentialsRequest
   metadata:
     name: <component_credentials_request>
     namespace: openshift-cloud-credential-operator
     ...
   spec:
   ```

```
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
  ...
```

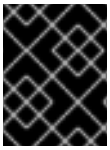
5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

7.7.8.2. 短期認証情報を使用するように Azure クラスターを設定する

Microsoft Entra Workload ID を使用するクラスターをインストールするには、Cloud Credential Operator ユーティリティーを設定し、クラスターに必要な Azure リソースを作成する必要があります。

7.7.8.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- 次の権限で使用する **ccoctl** ユーティリティー用のグローバル Microsoft Azure アカウントが作成されました。

  例7.32 必要な Azure 権限

  - Microsoft.Resources/subscriptions/resourceGroups/read
  - Microsoft.Resources/subscriptions/resourceGroups/write
  - Microsoft.Resources/subscriptions/resourceGroups/delete
  - Microsoft.Authorization/roleAssignments/read
  - Microsoft.Authorization/roleAssignments/delete
  - Microsoft.Authorization/roleAssignments/write
  - Microsoft.Authorization/roleDefinitions/read
  - Microsoft.Authorization/roleDefinitions/write
  - Microsoft.Authorization/roleDefinitions/delete
  - Microsoft.Storage/storageAccounts/listkeys/action
  - Microsoft.Storage/storageAccounts/delete
  - Microsoft.Storage/storageAccounts/read
  - Microsoft.Storage/storageAccounts/write
  - Microsoft.Storage/storageAccounts/blobServices/containers/write
  - Microsoft.Storage/storageAccounts/blobServices/containers/delete
  - Microsoft.Storage/storageAccounts/blobServices/containers/read

- Microsoft.ManagedIdentity/userAssignedIdentities/delete
- Microsoft.ManagedIdentity/userAssignedIdentities/read
- Microsoft.ManagedIdentity/userAssignedIdentities/write
- Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/read
- Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/write
- Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/delete
- Microsoft.Storage/register/action
- Microsoft.ManagedIdentity/register/action

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

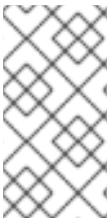
```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

7.7.8.2.2. Cloud Credential Operator ユーティリティーを使用した Azure リソースの作成

ccoctl azure create-all コマンドを使用して、Azure リソースの作成を自動化できます。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。
- Azure CLI を使用して Microsoft Azure アカウントにアクセスします。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. **ccoctl** ユーティリティーが Azure 認証情報を自動的に検出できるようにするには、次のコマンドを実行して Azure CLI にログインします。

```
$ az login
```

4. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl azure create-all \
  --name=<azure_infra_name> \ 1
  --output-dir=<ccoctl_output_dir> \ 2
  --region=<azure_region> \ 3
  --subscription-id=<azure_subscription_id> \ 4
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 5
  --dnszone-resource-group-name=<azure_dns_zone_resource_group_name> \ 6
  --tenant-id=<azure_tenant_id> 7
```

1. トラッキングに使用される、作成されたすべての Azure リソースのユーザー定義名を指定します。
2. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
3. クラウドリソースが作成される Azure リージョンです。
4. 使用する Azure サブスクリプション ID を指定します。
5. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。

6 クラスターのベースドメイン Azure DNS ゾーンを含むリソースグループの名前を指定します。

7 使用する Azure テナント ID を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

追加のオプションパラメーターとその使用方法の説明を表示するには、**azure create-all --help** コマンドを実行します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
azure-ad-pod-identity-webhook-config.yaml
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-azure-cloud-credentials-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capz-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-disk-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-file-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-azure-cloud-credentials-credentials.yaml
```

Azure をクエリーすることで、Microsoft Entra ID サービスアカウントが作成されていることを確認できます。詳細は、Entra ID サービスアカウントのリストに関する Azure ドキュメントを参照してください。

7.7.8.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. 既存のリソースグループを使用する代わりに、**ccoctl** ユーティリティーを使用して新しい Azure リソースグループを作成した場合は、次のように **install-config.yaml** の **resourceGroupName** パラメーターを変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   # ...
   platform:
     azure:
       resourceGroupName: <azure_infra_name> 1
   # ...
   ```

   1 この値は、**ccoctl azure create-all** コマンドの **--name** 引数で指定された Azure リソースのユーザー定義名と一致する必要があります。

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

7.7.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

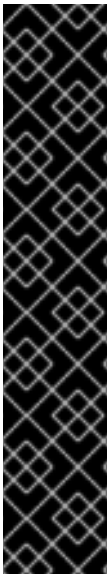
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

7.7.10. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.7.11. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

7.8. プライベートクラスターの AZURE へのインストール

OpenShift Container Platform バージョン 4.16 では、Microsoft Azure 上の既存の Azure Virtual Network (VNet) にプライベートクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

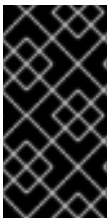
7.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- 暗号化のために Azure 環境を準備 した (顧客管理の暗号化キーを使用する場合)。

7.8.2. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。



重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。
- 以下にアクセスできるマシンからデプロイ。
  - プロビジョニングするクラウドの API サービス。
  - プロビジョニングするネットワーク上のホスト。
  - インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

7.8.2.1. Azure のプライベートクラスター

Microsoft Azure でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VNet とサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

ネットワークがプライベート VNET に接続される方法によって、クラスターのプライベート DNS レコードを解決するために DNS フォワーダーを使用する必要がある場合があります。クラスターのマシ

ンは、DNS 解決に **168.63.129.16** を内部で使用します。詳細は、Azure ドキュメントの What is Azure Private DNS? および What is IP address 168.63.129.16? を参照してください。

クラスターには、Azure API にアクセスするためにインターネットへのアクセスが依然として必要です。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- **BaseDomainResourceGroup** (クラスターがパブリックレコードを作成しないため)
- パブリック IP アドレス
- パブリック DNS レコード
- パブリックエンドポイント

> The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

7.8.2.1.1. 制限事項

Azure 上のプライベートクラスターは、既存の VNet の使用に関連する制限のみの制限を受けます。

7.8.2.2. ユーザー定義のアウトバウンドルーティング

OpenShift Container Platform では、クラスターがインターネットに接続するために独自のアウトバウンドルーティングを選択できます。これにより、パブリック IP アドレスおよびパブリックロードバランサーの作成を省略できます。

クラスターをインストールする前に、**install-config.yaml** ファイルのパラメーターを変更してユーザー定義のルーティングを設定できます。クラスターのインストール時にアウトバウンドルーティングを使用するには、既存の VNet が必要です。インストールプログラムはこれを設定しません。

クラスターをユーザー定義のルーティングを使用するように設定する際に、インストールプログラムは以下のリソースを作成しません。

- インターネットにアクセスするためのアウトバウンドルール。
- パブリックロードバランサーのパブリック IP。
- アウトバウンド要求のパブリックロードバランサーにクラスターマシンを追加する Kubernetes Service オブジェクト。

ユーザー定義のルーティングを設定する前に、以下の項目が利用可能であることを確認する必要があります。

- OpenShift イメージレジストリーミラーを使用しない場合は、コンテナーイメージのプルにインターネットへの egress を使用できます。
- クラスターは Azure API にアクセスできます。
- 各種の allowlist エンドポイントが設定されます。これらのエンドポイントについては、**ファイアウォールの設定**セクションで参照できます。

ユーザー定義のルーティングを使用したインターネットアクセスでサポートされる既存のネットワーク設定がいくつかあります。

ネットワークアドレス変換のあるプライベートクラスター
Azure VNET NAT (ネットワークアドレス変換) を使用して、クラスター内のサブネットのアウトバウンドインターネットアクセスを提供できます。設定手順については、Azure ドキュメントの Create a NAT gateway using Azure CLI を参照してください。

Azure NAT およびユーザー定義のルーティングが設定された VNet 設定を使用する場合、パブリックエンドポイントのないプライベートクラスターを作成できます。

Azure ファイアウォールのあるプライベートクラスター
Azure ファイアウォールを使用して、クラスターのインストールに使用される VNet のアウトバウンドルーティングを提供できます。Azure ドキュメントで Azure ファイアウォールのあるユーザー定義のルーティングを提供する方法 について確認することができます。

Azure ファイアウォールおよびユーザー定義のルーティングが設定された VNet 設定を使用する場合、パブリックエンドポイントのないプライベートクラスターを作成できます。

プロキシー設定のあるプライベートクラスター
ユーザー定義のルーティングと共にプロキシーを使用し、インターネットへの egress を許可することができます。クラスター Operator がプロキシーを使用して Azure API にアクセスしないようにする必要があります。Operator はプロキシー外から Azure API にアクセスできる必要があります。

0.0.0.0/0 が Azure によって自動的に設定された状態で、サブネットのデフォルトのルートテーブルを使用する場合、すべての Azure API 要求は、IP アドレスがパブリックな場合でも Azure の内部ネットワークでルーティングされます。ネットワークセキュリティーグループのルールが Azure API エンドポイントへの egress を許可している限り、ユーザー定義のルーティングが設定されたプロキシーにより、パブリックエンドポイントのないプライベートクラスターを作成できます。

インターネットアクセスのないプライベートクラスター
Azure API を除く、インターネットへのすべてのアクセスを制限するプライベートネットワークをインストールできます。これは、リリースイメージレジストリーをローカルにミラーリングすることによって実現されます。クラスターは以下にアクセスできる必要があります。

- コンテナーイメージのプルを可能にする OpenShift イメージレジストリーミラー
- Azure API へのアクセス

各種の要件を利用可能な場合、ユーザー定義のルーティングを使用して、パブリックエンドポイントのないプライベートクラスターを作成できます。

7.8.3. OpenShift Container Platform クラスターでの VNet の再利用について

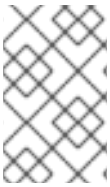
OpenShift Container Platform 4.16 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

7.8.3.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加のネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。

注記

デフォルトでは、**install-config.yaml** ファイルでアベイラビリティゾーンを指定すると、インストールプログラムはコントロールプレーンマシンとコンピューティングマシンを リージョン 内の これらのアベイラビリティゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定されたサブネットがすべて存在します。

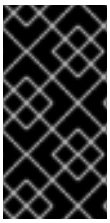
- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットがあります
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

注記

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

7.8.3.1.1. ネットワークセキュリティーグループの要件

コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。



重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表7.16 必須ポート

ポート	説明	コントロールプレーン	Compute
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	
22623	マシンをプロビジョニングするためのマシン設定サーバーへの内部通信を許可します。	x	

1. Azure Firewall を使用してインターネットアクセスを制限している場合は、Azure API を許可するように Azure Firewall を設定できます。ネットワークセキュリティーグループルールは必要ありません。

重要

現在、マシン設定サーバーエンドポイントをブロックまたは制限する方法はサポートされていません。マシン設定サーバーは、既存の設定または状態を持たない新しくプロビジョニングされたマシンが設定を取得できるように、ネットワークに公開する必要があります。このモデルでは、信頼のルートは証明書署名要求 (CSR) エンドポイントであり、kubelet がクラスターに参加するための承認のために証明書署名要求を送信する場所です。このため、シークレットや証明書などの機密情報を配布するためにマシン設定を使用しないでください。

マシン設定サーバーエンドポイント、ポート 22623 および 22624 がベアメタルシナリオで確実に保護されるようにするには、顧客は適切なネットワークポリシーを設定する必要があります。

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

関連情報

- OpenShift SDN ネットワークプラグインについて
- ファイアウォールの設定

7.8.3.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

7.8.3.3. クラスター間の分離

クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

7.8.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.8.5. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

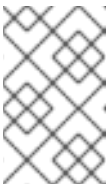
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.8.6. インストールプログラムの取得

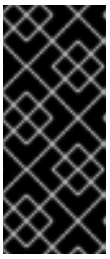
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

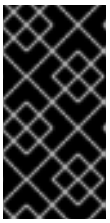
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.8.7. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- Azure のインストール設定パラメーター

7.8.7.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表7.17 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

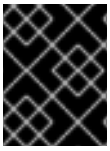
3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。



重要

premiumIO パラメーターが **true** に設定されている Azure 仮想マシンを使用する必要があります。

フラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

7.8.7.2. Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure インスタンスタイプは OpenShift Container Platform でテストされています。

例7.33 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

7.8.7.3. 64 ビット ARM インフラストラクチャー上の Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure Azure64 インスタンスタイプは OpenShift Container Platform でテストされています。

例7.34 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

7.8.7.4. Azure VM の信頼された起動の有効化

Azure にクラスターをインストールするときに、セキュアブート と 仮想化された信頼できるプラットフォームモジュール という 2 つの信頼された起動機能を有効にできます。

これらの機能をサポートする 仮想マシンのサイズ については、仮想マシンのサイズに関する Azure のドキュメントを参照してください。

重要

信頼できる起動はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: TrustedLaunch 2
        trustedLaunch:
          uefiSettings:
            secureBoot: Enabled 3
            virtualizedTrustedPlatformModule: Enabled 4
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定すると、それぞれコントロールプレーンまたはコンピュートノードのみで信頼された起動が有効になります。すべてのノードで信頼できる起動を有効にするには、**platform.azure.defaultMachinePlatform** を指定します。
2. 信頼できる起動機能を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。

7.8.7.5. Confidential VM の有効化

クラスターをインストールするときに、Confidential 仮想マシンを有効にできます。コンピューティングノード、コンピュートノード、またはすべてのノードに対して Confidential 仮想マシンを有効にできます。

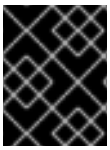
重要

Confidential VMs の使用はテクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

次の仮想マシンサイズの Confidential VM を使用できます。

- DCasv5 シリーズ
- DCadsv5 シリーズ
- ECasv5 シリーズ
- ECadsv5 シリーズ



重要

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

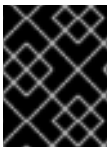
- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: ConfidentialVM 2
        confidentialVM:
          uefiSettings:
            secureBoot: Enabled 3
            virtualizedTrustedPlatformModule: Enabled 4
      osDisk:
        securityProfile:
          securityEncryptionType: VMGuestStateOnly 5
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定して、それぞれコントロールプレーンまたはコンピュートノードのみに Confidential 仮想マシンをデプロイし
2. Confidential VM を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。
5. 仮想マシンゲストの状態を暗号化するには、**VMGuestStateOnly** を指定します。

7.8.7.6. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      encryptionAtHost: true
      ultraSSDCapability: Enabled
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      ultraSSDCapability: Enabled
      type: Standard_D2s_v3
```

```
      encryptionAtHost: true
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 11
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    defaultMachinePlatform:
      osImage: 12
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      ultraSSDCapability: Enabled
    baseDomainResourceGroupName: resource_group 13
    region: centralus 14
    resourceGroupName: existing_resource_group 15
    networkResourceGroupName: vnet_resource_group 16
    virtualNetwork: vnet 17
    controlPlaneSubnet: control_plane_subnet 18
    computeSubnet: compute_subnet 19
    outboundType: UserDefinedRouting 20
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 21
fips: false 22
sshKey: ssh-ed25519 AAAA... 23
publish: Internal 24
```

1 10 14 21 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンのリストを指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 オプション: コントロールプレーンとコンピュートマシンを起動するために使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) イメージ。**platform.azure.defaultMachinePlatform.osImage** の下の **publisher**、**offer**、**sku**、および **version** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**controlPlane.platform.azure.osImage** または **compute.platform.azure.osImage** の下のパラメーターが設定されている場合、それらは **platform.azure.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

13 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

15 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

16 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

17 既存の VNet を使用する場合は、その名前を指定します。

18 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

19 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

20 独自のアウトバウンドルーティングをカスタマイズすることができます。ユーザー定義のルーティングを設定すると、クラスターに外部エンドポイントが公開されなくなります。エグレスのユーザー定義のルーティングでは、クラスターを既存の VNet にデプロイする必要があります。

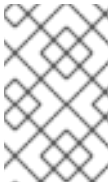
22 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

23 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

24 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

7.8.7.7. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

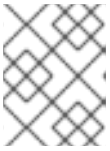
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

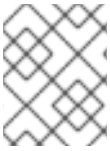
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

関連情報

- 高速ネットワークの詳細については、Accelerated Networking for Microsoft Azure VMs を参照してください。

7.8.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。

5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.8.9. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように Azure クラスターを設定する の手順に従ってください。

7.8.9.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
    ...
```

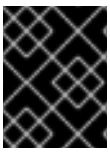
5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```



重要

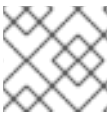
手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

7.8.9.2. 短期認証情報を使用するように Azure クラスターを設定する

Microsoft Entra Workload ID を使用するクラスターをインストールするには、Cloud Credential Operator ユーティリティーを設定し、クラスターに必要な Azure リソースを作成する必要があります。

7.8.9.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- 次の権限で使用する **ccoctl** ユーティリティー用のグローバル Microsoft Azure アカウントが作成されました。

  例7.35 必要な Azure 権限

  - Microsoft.Resources/subscriptions/resourceGroups/read
  - Microsoft.Resources/subscriptions/resourceGroups/write
  - Microsoft.Resources/subscriptions/resourceGroups/delete
  - Microsoft.Authorization/roleAssignments/read
  - Microsoft.Authorization/roleAssignments/delete
  - Microsoft.Authorization/roleAssignments/write
  - Microsoft.Authorization/roleDefinitions/read
  - Microsoft.Authorization/roleDefinitions/write
  - Microsoft.Authorization/roleDefinitions/delete
  - Microsoft.Storage/storageAccounts/listkeys/action
  - Microsoft.Storage/storageAccounts/delete
  - Microsoft.Storage/storageAccounts/read
  - Microsoft.Storage/storageAccounts/write
  - Microsoft.Storage/storageAccounts/blobServices/containers/write
  - Microsoft.Storage/storageAccounts/blobServices/containers/delete
  - Microsoft.Storage/storageAccounts/blobServices/containers/read
  - Microsoft.ManagedIdentity/userAssignedIdentities/delete
  - Microsoft.ManagedIdentity/userAssignedIdentities/read
  - Microsoft.ManagedIdentity/userAssignedIdentities/write
  - Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/read
  - Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/write
  - Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/delete
  - Microsoft.Storage/register/action
  - Microsoft.ManagedIdentity/register/action

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

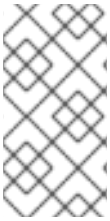
ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
```

```
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

7.8.9.2.2. Cloud Credential Operator ユーティリティーを使用した Azure リソースの作成

ccoctl azure create-all コマンドを使用して、Azure リソースの作成を自動化できます。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。
- Azure CLI を使用して Microsoft Azure アカウントにアクセスします。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

指したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. **ccoctl** ユーティリティーが Azure 認証情報を自動的に検出できるようにするには、次のコマンドを実行して Azure CLI にログインします。

```
$ az login
```

4. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl azure create-all \
  --name=<azure_infra_name> \ 1
  --output-dir=<ccoctl_output_dir> \ 2
  --region=<azure_region> \ 3
  --subscription-id=<azure_subscription_id> \ 4
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 5
  --dnszone-resource-group-name=<azure_dns_zone_resource_group_name> \ 6
  --tenant-id=<azure_tenant_id> 7
```

1. トラッキングに使用される、作成されたすべての Azure リソースのユーザー定義名を指定します。
2. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
3. クラウドリソースが作成される Azure リージョンです。
4. 使用する Azure サブスクリプション ID を指定します。
5. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
6. クラスターのベースドメイン Azure DNS ゾーンを含むリソースグループの名前を指定します。
7. 使用する Azure テナント ID を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

追加のオプションパラメーターとその使用方法の説明を表示するには、**azure create-all --help** コマンドを実行します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
azure-ad-pod-identity-webhook-config.yaml
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-azure-cloud-credentials-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capz-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-disk-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-file-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-azure-cloud-credentials-credentials.yaml
```

Azure をクエリーすることで、Microsoft Entra ID サービスアカウントが作成されていることを確認できます。詳細は、Entra ID サービスアカウントのリストに関する Azure ドキュメントを参照してください。

7.8.9.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

2. 既存のリソースグループを使用する代わりに、**ccoctl** ユーティリティーを使用して新しい Azure リソースグループを作成した場合は、次のように **install-config.yaml** の **resourceGroupName** パラメーターを変更します。

   設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
# ...
platform:
  azure:
    resourceGroupName: <azure_infra_name> 1
# ...
```

1 この値は、**ccoctl azure create-all** コマンドの **--name** 引数で指定された Azure リソースのユーザー定義名と一致する必要があります。

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

7.8.10. オプション: プライベートイメージレジストリー用のプライベート Microsoft Azure クラスターを準備する

プライベートイメージレジストリーをプライベート Microsoft Azure クラスターにインストールすることで、プライベートストレージのエンドポイントを作成できます。プライベートストレージのエンドポイントは、レジストリーのストレージアカウントに対する公開エンドポイントを無効にし、OpenShift Container Platform デプロイメントにセキュリティーのレイヤーを追加します。次のガイドに従い、プライベートイメージレジストリーを使用してインストールするためのプライベート Microsoft Azure クラスターを準備します。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (oc) がインストールされている。
- 以下の情報を含む **install-config.yaml** の準備が完了している。
  - **publish** フィールドは **Internal** に設定されている。
- プライベートストレージエンドポイントの作成権限を設定している。詳細は、「installer-provisioned infrastructure の Azure 権限」を参照してください。

手順

1. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

このコマンドを実行すると、以下の情報が表示されます。

出力例

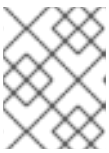
```
INFO Consuming Install Config from target directory
INFO Manifests created in: <installation_directory>/manifests and
<installation_directory>/openshift
```

2. イメージレジストリー設定オブジェクトを作成し、Microsoft Azure から提供された **networkResourceGroupName**、**subnetName**、**vnetName** を渡します。以下に例を示します。

```
$ touch imageregistry-config.yaml
```

```
apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  name: cluster
spec:
  managementState: "Managed"
  replicas: 2
  rolloutStrategy: RollingUpdate
  storage:
    azure:
      networkAccess:
        internal:
          networkResourceGroupName: <vnet_resource_group> 1
          subnetName: <subnet_name> 2
          vnetName: <vnet_name> 3
        type: Internal
```

1. オプション: 既存の VNet およびサブネット設定がある場合は、**<vnet_resource_group>** を既存の仮想ネットワーク (VNet) を含むリソースグループ名に置き換えます。
2. オプション: 既存の VNet およびサブネット設定がある場合は、**<subnet_name>** を指定されたリソースグループ内にある既存のコンピュートサブネットの名前に置き換えます。
3. オプション: 既存の VNet およびサブネット設定がある場合は、**<vnet_name>** を指定されたリソースグループ内にある既存の仮想ネットワーク (VNet) の名前に置き換えます。



注記

imageregistry-config.yaml ファイルはインストールプロセス時に使用されます。必要に応じて、インストール前にバックアップする必要があります。

3. 次のコマンドを実行して、**imageregistry-config.yaml** ファイルを**<installation_directory/manifests>** フォルダーに移動します。

```
$ mv imageregistry-config.yaml <installation_directory/manifests/>
```

次のステップ

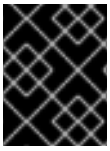
- **imageregistry-config.yaml** ファイルを **<installation_directory/manifests>** フォルダーに移動し、必要な権限を設定してから、クラスターのデプロイに進みます。

関連情報

- プライベートストレージエンドポイントの作成に必要な権限の一覧は、installer-provisioned infrastructure に必要な Azure 権限 を参照してください。

7.8.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルを使用してクラスターをインストールしている場合は、そのアプリケーション ID とパスワードが必要です。
- システムが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、インストールプログラムを実行する仮想マシン上でそれが有効になっています。
- ユーザーが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、次の前提条件を満たしている必要があります。
  - そのクライアント ID がある。
  - これは、インストールプログラムを実行する仮想マシンに割り当てられている。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルまたはマネージド ID を使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
   このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
```

```
--log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。

3. サブスクリプションの次の Azure パラメーター値を入力します。
   - **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
   - **azure tenant id** テナント ID を入力します。
4. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアント ID** の入力を求められたら、次のいずれかを行います。
   - サービスプリンシパルを使用している場合は、そのアプリケーション ID を入力します。
   - システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
   - ユーザー割り当てのマネージド ID を使用している場合は、そのクライアント ID を指定します。
5. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアントシークレット** の入力を求められたら、次のいずれかを実行します。
   - サービスプリンシパルを使用している場合は、そのパスワードを入力します。
   - システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
   - ユーザー割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの **~/.azure/** ディレクトリーに保存します。これにより、インストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作成するときにプロファイルをロードできるようになります。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

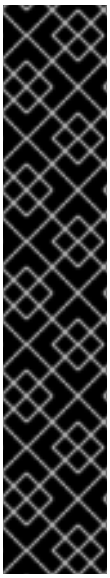
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

7.8.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

7.8.13. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.8.14. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

7.9. AZURE の GOVERNMENT リージョンへのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、Microsoft Azure のクラスターを government リージョンにインストールできます。government リージョンを設定するには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

7.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みの government リージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、長期間認証情報を手動で作成および維持 することができます。
- 暗号化のために Azure 環境を準備 した (顧客管理の暗号化キーを使用する場合)。

7.9.2. Azure government リージョン

OpenShift Container Platform は、クラスターの Microsoft Azure Government (MAG) リージョンへのデプロイをサポートします。MAG は、機密ワークロードを Azure で実行する必要のある連邦、州、地方の米国の各種の政府機関、請負業者、教育機関、およびその他の米国の顧客向けに設計されています。MAG は、government のみのデータセンターリージョンで設定されており、すべてに 影響レベル 5 の暫定認証 が付与されます。

MAG リージョンにインストールするには、**install-config.yaml** ファイルで Azure Government 専用のクラウドインスタンスおよびリージョンを手動で設定する必要があります。また、適切な government 環境を参照するようにサービスプリンシパルを更新する必要もあります。

注記

Azure government リージョンは、インストールプログラムからガイド付きターミナルプロンプトを使用して選択することはできません。リージョンは **install-config.yaml** ファイルで手動で定義する必要があります。指定されたリージョンに基づいて、**AzureUSGovernmentCloud** などの専用のクラウドインスタンスも設定するようにしてください。

7.9.3. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。
- 以下にアクセスできるマシンからデプロイ。
  - プロビジョニングするクラウドの API サービス。
  - プロビジョニングするネットワーク上のホスト。
  - インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

7.9.3.1. Azure のプライベートクラスター

Microsoft Azure でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VNet とサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

ネットワークがプライベート VNET に接続される方法によって、クラスターのプライベート DNS レコードを解決するために DNS フォワーダーを使用する必要がある場合があります。クラスターのマシンは、DNS 解決に **168.63.129.16** を内部で使用します。詳細は、Azure ドキュメントの What is Azure Private DNS? および What is IP address 168.63.129.16? を参照してください。

クラスターには、Azure API にアクセスするためにインターネットへのアクセスが依然として必要です。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- **BaseDomainResourceGroup** (クラスターがパブリックレコードを作成しないため)
- パブリック IP アドレス
- パブリック DNS レコード
- パブリックエンドポイント

> The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

7.9.3.1.1. 制限事項

Azure 上のプライベートクラスターは、既存の VNet の使用に関連する制限のみの制限を受けます。

7.9.3.2. ユーザー定義のアウトバウンドルーティング

OpenShift Container Platform では、クラスターがインターネットに接続するために独自のアウトバウンドルーティングを選択できます。これにより、パブリック IP アドレスおよびパブリックロードバランサーの作成を省略できます。

クラスターをインストールする前に、**install-config.yaml** ファイルのパラメーターを変更してユーザー定義のルーティングを設定できます。クラスターのインストール時にアウトバウンドルーティングを使用するには、既存の VNet が必要です。インストールプログラムはこれを設定しません。

クラスターをユーザー定義のルーティングを使用するように設定する際に、インストールプログラムは以下のリソースを作成しません。

- インターネットにアクセスするためのアウトバウンドルール。
- パブリックロードバランサーのパブリック IP。
- アウトバウンド要求のパブリックロードバランサーにクラスターマシンを追加する Kubernetes Service オブジェクト。

ユーザー定義のルーティングを設定する前に、以下の項目が利用可能であることを確認する必要があります。

- OpenShift イメージレジストリーミラーを使用しない場合は、コンテナーイメージのプルにインターネットへの egress を使用できます。
- クラスターは Azure API にアクセスできます。
- 各種の allowlist エンドポイントが設定されます。これらのエンドポイントについては、**ファイアウォールの設定**セクションで参照できます。

ユーザー定義のルーティングを使用したインターネットアクセスでサポートされる既存のネットワーク設定がいくつかあります。

7.9.4. OpenShift Container Platform クラスターでの VNet の再利用について

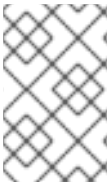
OpenShift Container Platform 4.16 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

7.9.4.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加のネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。

注記

デフォルトでは、**install-config.yaml** ファイルでアベイラビリティゾーンを指定すると、インストールプログラムはコントロールプレーンマシンとコンピューティングマシンを リージョン 内の これらのアベイラビリティゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定されたサブネットがすべて存在します。
- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットがあります
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。必要な場合に、インストールプログラムはコントロールプレーンおよびワーカーノードを管理するパブリックロードバランサーを作成し、Azure はパブリック IP アドレスをそれらに割り当てます。

注記

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

7.9.4.1.1. ネットワークセキュリティーグループの要件

コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。

重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表7.18 必須ポート

ポート	説明	コントロールプレーン	Compute
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	
22623	マシンをプロビジョニングするためのマシン設定サーバーへの内部通信を許可します。	x	

1. Azure Firewall を使用してインターネットアクセスを制限している場合は、Azure API を許可するように Azure Firewall を設定できます。ネットワークセキュリティーグループルールは必要ありません。

重要

現在、マシン設定サーバーエンドポイントをブロックまたは制限する方法はサポートされていません。マシン設定サーバーは、既存の設定または状態を持たない新しくプロビジョニングされたマシンが設定を取得できるように、ネットワークに公開する必要があります。このモデルでは、信頼のルートは証明書署名要求 (CSR) エンドポイントであり、kubelet がクラスターに参加するための承認のために証明書署名要求を送信する場所です。このため、シークレットや証明書などの機密情報を配布するためにマシン設定を使用しないでください。

マシン設定サーバーエンドポイント、ポート 22623 および 22624 がベアメタルシナリオで確実に保護されるようにするには、顧客は適切なネットワークポリシーを設定する必要があります。

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

関連情報

- OpenShift SDN ネットワークプラグインについて
- ファイアウォールの設定

7.9.4.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネ

ントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

7.9.4.3. クラスター間の分離

クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

7.9.5. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.9.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

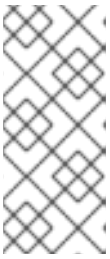
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

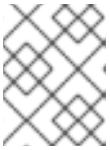
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

(1) ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.9.7. インストールプログラムの取得

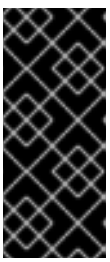
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

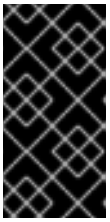
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.9.8. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

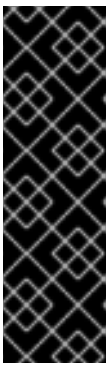
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- Azure のインストール設定パラメーター

7.9.8.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表7.19 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

重要

premiumIO パラメーターが **true** に設定されている Azure 仮想マシンを使用する必要があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

7.9.8.2. Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure インスタンスタイプは OpenShift Container Platform でテストされています。

例7.36 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***

- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

7.9.8.3. Azure VM の信頼された起動の有効化

Azure にクラスターをインストールするときに、セキュアブート と 仮想化された信頼できるプラットフォームモジュール という 2 つの信頼された起動機能を有効にできます。

これらの機能をサポートする 仮想マシンのサイズ については、仮想マシンのサイズに関する Azure のドキュメントを参照してください。

重要

信頼できる起動はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: TrustedLaunch 2
        trustedLaunch:
          uefiSettings:
            secureBoot: Enabled 3
            virtualizedTrustedPlatformModule: Enabled 4
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定すると、それぞれコントロールプレーンまたはコンピュートノードのみで信頼された起動が有効になります。すべてのノードで信頼できる起動を有効にするには、**platform.azure.defaultMachinePlatform** を指定します。
2. 信頼できる起動機能を有効にします。

3 Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。

4 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。

7.9.8.4. Confidential VM の有効化

クラスターをインストールするときに、Confidential 仮想マシンを有効にできます。コンピューティングノード、コンピュートノード、またはすべてのノードに対して Confidential 仮想マシンを有効にできます。

重要

Confidential VMs の使用はテクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

次の仮想マシンサイズの Confidential VM を使用できます。

- DCasv5 シリーズ
- DCadsv5 シリーズ
- ECasv5 シリーズ
- ECadsv5 シリーズ

重要

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

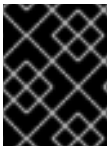
```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: ConfidentialVM 2
```

```
      confidentialVM:
        uefiSettings:
          secureBoot: Enabled (3)
          virtualizedTrustedPlatformModule: Enabled (4)
      osDisk:
        securityProfile:
          securityEncryptionType: VMGuestStateOnly (5)
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定して、それぞれコントロールプレーンまたはコンピュートノードのみに Confidential 仮想マシンをデプロイします。すべてのノードに Confidential 仮想マシンをデプロイするには、**platform.azure.defaultMachinePlatform** を指定します。
2. Confidential VM を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。
5. 仮想マシンゲストの状態を暗号化するには、**VMGuestStateOnly** を指定します。

7.9.8.5. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      encryptionAtHost: true
      ultraSSDCapability: Enabled
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
```

```
        version: example_image_version
      type: Standard_D8s_v3
  replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      ultraSSDCapability: Enabled
      type: Standard_D2s_v3
      encryptionAtHost: true
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 11
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    defaultMachinePlatform:
      osImage: 12
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      ultraSSDCapability: Enabled
    baseDomainResourceGroupName: resource_group 13
    region: usgovvirginia
    resourceGroupName: existing_resource_group 14
    networkResourceGroupName: vnet_resource_group 15
    virtualNetwork: vnet 16
    controlPlaneSubnet: control_plane_subnet 17
```

```
    computeSubnet: compute_subnet (18)
    outboundType: UserDefinedRouting (19)
    cloudName: AzureUSGovernmentCloud (20)
pullSecret: '{"auths": ...}' (21)
fips: false (22)
sshKey: ssh-ed25519 AAAA... (23)
publish: Internal (24)
```

(1) (10) (21) 必須。

(2) (6) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (7) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(4) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

(5) (8) 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

(9) マシンをデプロイするゾーンのリストを指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

(11) インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

(12) オプション: コントロールプレーンとコンピュートマシンを起動するために使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) イメージ。**platform.azure.defaultMachinePlatform.osImage** の下の **publisher**、**offer**、**sku**、および **version** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**controlPlane.platform.azure.osImage** または **compute.platform.azure.osImage** の下のパラメーターが設定されている場合、それらは **platform.azure.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

(13) ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

(14) クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

(15) 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

(16) 既存の VNet を使用する場合は、その名前を指定します。

17 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

18 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

19 独自のアウトバウンドルーティングをカスタマイズすることができます。ユーザー定義のルーティングを設定すると、クラスターに外部エンドポイントが公開されなくなります。エグレスのユーザー定義のルーティングでは、クラスターを既存の VNet にデプロイする必要があります。

20 クラスターをデプロイする Azure クラウド環境の名前を指定します。**AzureUSGovernmentCloud** を Microsoft Azure Government (MAG) リージョンにデプロイするように設定します。デフォルト値は **AzurePublicCloud** です。

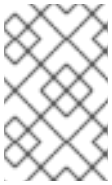
22 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

23 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

24 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

7.9.8.6. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-**

bundle 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

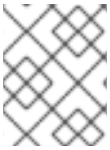
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

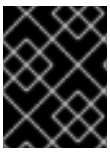
cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

関連情報

- 高速ネットワークの詳細については、Accelerated Networking for Microsoft Azure VMs を参照してください。

7.9.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルを使用してクラスターをインストールしている場合は、そのアプリケーション ID とパスワードが必要です。

- システムが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、インストールプログラムを実行する仮想マシン上でそれが有効になっています。
- ユーザーが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、次の前提条件を満たしている必要があります。
  - そのクライアント ID がある。
  - これは、インストールプログラムを実行する仮想マシンに割り当てられている。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルまたはマネージド ID を使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
   このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

   インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。
3. サブスクリプションの次の Azure パラメーター値を入力します。
   - **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
   - **azure tenant id** テナント ID を入力します。
4. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアント ID** の入力を求められたら、次のいずれかを行います。
   - サービスプリンシパルを使用している場合は、そのアプリケーション ID を入力します。
   - システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
   - ユーザー割り当てのマネージド ID を使用している場合は、そのクライアント ID を指定します。
5. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアントシークレット** の入力を求められたら、次のいずれかを実行します。
   - サービスプリンシパルを使用している場合は、そのパスワードを入力します。
   - システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

- ユーザー割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの ~/**.azure**/ ディレクトリーに保存します。これにより、インストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作成するときにプロファイルをロードできるようになります。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

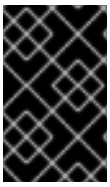
```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

7.9.10. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

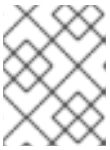
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.9.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

7.9.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.9.13. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

7.10. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での AWS へのクラスターのインストール

OpenShift Container Platform では、独自に提供するインフラストラクチャーを使用して、Microsoft Azure にクラスターをインストールできます。

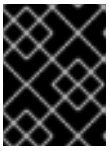
これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Azure Resource Manager (ARM) テンプレートが提供されます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の ARM テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。

前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了する必要があります。

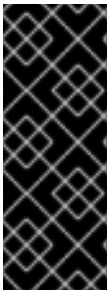
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- ご使用の環境でクラウド ID およびアクセス管理 (IAM) API にアクセスできない場合、または管理者レベルの認証情報シークレットを **kube-system** namespace に保存しない場合は、長期認証情報を手動で作成している。
- 暗号化のために Azure 環境を準備 した (顧客管理の暗号化キーを使用する場合)。

7.10.1. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

7.10.1.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

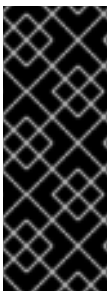
- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

7.10.1.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.10.2. Azure プロジェクトの設定

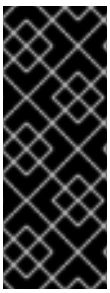
OpenShift Container Platform をインストールする前に、これをホストするために Azure プロジェクトを設定する必要があります。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

7.10.2.1. Azure アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure コンポーネントを使用し、デフォルトの Azure サブスクリプションおよびサービス制限、クォータ、および制約 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。



重要

デフォルトの制限は、Free Trial や Pay-As-You-Go、および DV2、F、および G などのシリーズといったカテゴリータイプによって異なります。たとえば、Enterprise Agreement サブスクリプションのデフォルトは 350 コアです。

サブスクリプションタイプの制限を確認し、必要に応じて、デフォルトのクラスターを Azure にインストールする前にアカウントのクォータ制限を引き上げます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
vCPU	44	リージョンごとに 20	デフォルトのクラスターには 44 vCPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップおよびコントロールプレーンマシンは 8 vCPU を使用する **Standard_D8s_v3** 仮想マシンを使用し、コンピュートマシンは 4 vCPU を使用する **Standard_D4s_v3** 仮想マシンを使用するため、 デフォルトのクラスターには 44 vCPU が必要です。8 vCPU を使用するブートストラップノードの仮想マシンは、インストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
OS ディスク	7		各クラスターマシンには、少なくとも 100 GB のストレージと 300 IOPS が必要です。これらはサポートされる最小の値ですが、実稼働クラスターおよび高負荷ワークロードがあるクラスターには、さらに高速なストレージが推奨されます。パフォーマンスを向上させるためのストレージ最適化について、詳しくは「スケーラビリティーとパフォーマンス」セクションの「ストレージの最適化」を参照してください。
VNet	1	リージョンごとに 1000	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。
ネットワークインターフェイス	7	リージョンごとに 65,536	各デフォルトクラスターには、7 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。
ネットワークセキュリティーグループ	2	5000	各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
ネットワークロードバランサー	3	リージョンごとに 1000	各クラスターは以下の ロードバランサー を作成します。 **default**: ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス **internal**: コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス **external**: コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス アプリケーションが追加の Kubernetes **LoadBalancer** サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。
パブリック IP アドレス	3		2 つのパブリックロードバランサーのそれぞれはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。
プライベート IP アドレス	7		内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。
スポット VM vCPU (オプション)	0 スポット VM を設定する場合には、クラスターのコンピュートノードごとにスポット VM vCPU が 2 つ必要です。	リージョンごとに 20	これはオプションのコンポーネントです。スポット VM を使用するには、Azure のデフォルトの制限を最低でも、クラスター内のコンピュートノード数の 2 倍に増やす必要があります。  **注記** コントロールプレーンノードにスポット VM を使用することは推奨しません。

関連情報

- ストレージの最適化

7.10.2.2. Azure でのパブリック DNS ゾーンの設定

OpenShift Container Platform をインストールするには、使用する Microsoft Azure アカウントに、専用のパブリックホスト DNS ゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。このサービスは、クラスターへの外部接続のためのクラスター DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、Azure または別のソースから新規のものを取得できます。

注記

Azure 経由でドメインを購入する方法についての詳細は、Azure ドキュメントの Buy a custom domain name for Azure App Service を参照してください。

2. 既存のドメインおよびレジストラーを使用している場合、その DNS を Azure に移行します。Azure ドキュメントの Migrate an active DNS name to Azure App Service を参照してください。

3. ドメインの DNS を設定します。Azure ドキュメントの Tutorial: Host your domain in Azure DNS の手順に従い、ドメインまたはサブドメインのパブリックホストゾーンを作成し、 新規の権威ネームサーバーを抽出し、ドメインが使用するネームサーバーのレジストラーレコードを更新します。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。

この DNS ゾーンの作成例 を参照し、Azure の DNS ソリューションを確認することができます。

7.10.2.3. Azure アカウント制限の拡張

アカウントの制限を引き上げるには、Azure ポータルでサポートをリクエストします。

注記

サポートリクエストごとに 1 つの種類のクォータのみを増やすことができます。

手順

1. Azure ポータルの左端で **Help + support** をクリックします。

2. **New support request** をクリックしてから必要な値を選択します。

   a. **Issue type** リストから、**Service and subscription limits (quotas)** を選択します。

   b. **Subscription** リストから、変更するサブスクリプションを選択します。

c. **Quota type** リストから、引き上げるクォータを選択します。たとえば、**Compute-VM (cores-vCPUs) subscription limit increases** を選択し、クラスターのインストールに必要な vCPU の数を増やします。

d. **Next: Solutions** をクリックします。

3. **Problem Details**ページで、クォータの引き上げについての必要な情報を指定します。

a. **Provide details**をクリックし、**Quota details**ウィンドウに必要な詳細情報を指定します。

b. SUPPORT METHOD and CONTACT INFO セクションに、問題の重大度および問い合わせ先の詳細を指定します。

4. **Next: Review + create** をクリックしてから **Create** をクリックします。

7.10.2.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

7.10.2.5. 必要な Azure ロール

OpenShift Container Platform クラスターでは、Azure リソースを作成および管理するために Azure ID が必要です。ID を作成する前に、環境が次の要件を満たしていることを確認してください。

- ID の作成に使用する Azure アカウントには、**User Access Administrator** ロールと **Contributor** ロールが割り当てられます。これらのロールは次の場合に必要です。
  - サービスプリンシパルまたはユーザー割り当てのマネージド ID を作成します。
  - 仮想マシン上でシステム割り当てマネージド ID を有効にします。
- サービスプリンシパルを使用してインストールを完了する場合は、ID の作成に使用する Azure アカウントに、Microsoft Entra ID の **Microsoft.directory/servicePrincipals/createAsOwner** パーミッションが割り当てられていることを確認してください。

Azure ポータルでロールを設定するには、Azure ドキュメントの Manage access to Azure resources using RBAC and the Azure portal を参照します。

7.10.2.6. ユーザーがプロビジョニングするインフラストラクチャーに必要な Azure アクセス許可

インストールプログラムでは、クラスターをデプロイし、日常の操作を維持するために必要なパーミッションを持つ Azure サービスプリンシパルまたはマネージド ID にアクセスする必要があります。これらパーミッションは、ID に関連付けられた Azure サブスクリプションに付与する必要があります。

以下のオプションを使用できます。

- ID に **Contributor** ロールと **User Access Administrator** ロールを割り当てることができます。これらのロールを割り当てるのが、必要な権限をすべて付与する最も簡単な方法です。
  ロールの割り当ての詳細は、Azure portal を使用した Azure リソースへのアクセスの管理 に関する Azure ドキュメントを参照してください。

- 組織のセキュリティーポリシーで、より制限的なアクセス許可のセットが必要な場合は、必要なアクセス許可を持つ カスタムロール を作成できます。

Microsoft Azure で OpenShift Container Platform クラスターを作成するには、以下のアクセス許可が必要です。

例7.37 承認リソースを作成するために必要な権限

- **Microsoft.Authorization/policies/audit/action**
- **Microsoft.Authorization/policies/auditIfNotExists/action**
- **Microsoft.Authorization/roleAssignments/read**
- **Microsoft.Authorization/roleAssignments/write**

例7.38 コンピューティングリソースの作成に必要な権限

- **Microsoft.Compute/images/read**
- **Microsoft.Compute/images/write**
- **Microsoft.Compute/images/delete**
- **Microsoft.Compute/availabilitySets/read**
- **Microsoft.Compute/disks/beginGetAccess/action**
- **Microsoft.Compute/disks/delete**
- **Microsoft.Compute/disks/read**
- **Microsoft.Compute/disks/write**
- **Microsoft.Compute/galleries/images/read**
- **Microsoft.Compute/galleries/images/versions/read**
- **Microsoft.Compute/galleries/images/versions/write**
- **Microsoft.Compute/galleries/images/write**
- **Microsoft.Compute/galleries/read**
- **Microsoft.Compute/galleries/write**
- **Microsoft.Compute/snapshots/read**
- **Microsoft.Compute/snapshots/write**
- **Microsoft.Compute/snapshots/delete**
- **Microsoft.Compute/virtualMachines/delete**
- **Microsoft.Compute/virtualMachines/powerOff/action**

- **Microsoft.Compute/virtualMachines/read**
- **Microsoft.Compute/virtualMachines/write**
- **Microsoft.Compute/virtualMachines/deallocate/action**

例7.39 ID 管理リソースを作成するために必要なアクセス許可

- **Microsoft.ManagedIdentity/userAssignedIdentities/assign/action**
- **Microsoft.ManagedIdentity/userAssignedIdentities/read**
- **Microsoft.ManagedIdentity/userAssignedIdentities/write**

例7.40 ネットワークリソースの作成に必要な権限

- **Microsoft.Network/dnsZones/A/write**
- **Microsoft.Network/dnsZones/CNAME/write**
- **Microsoft.Network/dnszones/CNAME/read**
- **Microsoft.Network/dnszones/read**
- **Microsoft.Network/loadBalancers/backendAddressPools/join/action**
- **Microsoft.Network/loadBalancers/backendAddressPools/read**
- **Microsoft.Network/loadBalancers/backendAddressPools/write**
- **Microsoft.Network/loadBalancers/read**
- **Microsoft.Network/loadBalancers/write**
- **Microsoft.Network/networkInterfaces/delete**
- **Microsoft.Network/networkInterfaces/join/action**
- **Microsoft.Network/networkInterfaces/read**
- **Microsoft.Network/networkInterfaces/write**
- **Microsoft.Network/networkSecurityGroups/join/action**
- **Microsoft.Network/networkSecurityGroups/read**
- **Microsoft.Network/networkSecurityGroups/securityRules/delete**
- **Microsoft.Network/networkSecurityGroups/securityRules/read**
- **Microsoft.Network/networkSecurityGroups/securityRules/write**
- **Microsoft.Network/networkSecurityGroups/write**
- **Microsoft.Network/privateDnsZones/A/read**

- Microsoft.Network/privateDnsZones/A/write
- Microsoft.Network/privateDnsZones/A/delete
- Microsoft.Network/privateDnsZones/SOA/read
- Microsoft.Network/privateDnsZones/read
- Microsoft.Network/privateDnsZones/virtualNetworkLinks/read
- Microsoft.Network/privateDnsZones/virtualNetworkLinks/write
- Microsoft.Network/privateDnsZones/write
- Microsoft.Network/publicIPAddresses/delete
- Microsoft.Network/publicIPAddresses/join/action
- Microsoft.Network/publicIPAddresses/read
- Microsoft.Network/publicIPAddresses/write
- Microsoft.Network/virtualNetworks/join/action
- Microsoft.Network/virtualNetworks/read
- Microsoft.Network/virtualNetworks/subnets/join/action
- Microsoft.Network/virtualNetworks/subnets/read
- Microsoft.Network/virtualNetworks/subnets/write
- Microsoft.Network/virtualNetworks/write

例7.41 リソースの正常性をチェックするために必要なアクセス許可

- Microsoft.Resourcehealth/healthevent/Activated/action
- Microsoft.Resourcehealth/healthevent/InProgress/action
- Microsoft.Resourcehealth/healthevent/Pending/action
- Microsoft.Resourcehealth/healthevent/Resolved/action
- Microsoft.Resourcehealth/healthevent/Updated/action

例7.42 リソースグループの作成に必要なアクセス許可

- Microsoft.Resources/subscriptions/resourceGroups/read
- Microsoft.Resources/subscriptions/resourcegroups/write

例7.43 リソースタグの作成に必要なアクセス許可

- **Microsoft.Resources/tags/write**

例7.44 ストレージリソースの作成に必要な権限

- **Microsoft.Storage/storageAccounts/blobServices/read**
- **Microsoft.Storage/storageAccounts/blobServices/containers/write**
- **Microsoft.Storage/storageAccounts/fileServices/read**
- **Microsoft.Storage/storageAccounts/fileServices/shares/read**
- **Microsoft.Storage/storageAccounts/fileServices/shares/write**
- **Microsoft.Storage/storageAccounts/fileServices/shares/delete**
- **Microsoft.Storage/storageAccounts/listKeys/action**
- **Microsoft.Storage/storageAccounts/read**
- **Microsoft.Storage/storageAccounts/write**

例7.45 デプロイメントの作成に必要な権限

- **Microsoft.Resources/deployments/read**
- **Microsoft.Resources/deployments/write**
- **Microsoft.Resources/deployments/validate/action**
- **Microsoft.Resources/deployments/operationstatuses/read**

例7.46 コンピュートリソースを作成するためのオプションのアクセス許可

- **Microsoft.Compute/availabilitySets/write**

例7.47 Marketplace 仮想マシンリソースを作成するためのオプションのアクセス許可

- **Microsoft.MarketplaceOrdering/offertypes/publishers/offers/plans/agreements/read**
- **Microsoft.MarketplaceOrdering/offertypes/publishers/offers/plans/agreements/write**

例7.48 ユーザー管理の暗号化を有効にするためのオプションのアクセス許可

- **Microsoft.Compute/diskEncryptionSets/read**
- **Microsoft.Compute/diskEncryptionSets/write**
- **Microsoft.Compute/diskEncryptionSets/delete**

- **Microsoft.KeyVault/vaults/read**
- **Microsoft.KeyVault/vaults/write**
- **Microsoft.KeyVault/vaults/delete**
- **Microsoft.KeyVault/vaults/deploy/action**
- **Microsoft.KeyVault/vaults/keys/read**
- **Microsoft.KeyVault/vaults/keys/write**
- **Microsoft.Features/providers/features/register/action**

Microsoft Azure で OpenShift Container Platform クラスターを削除するには、以下のアクセス許可が必要です。

例7.49 承認リソースを削除するために必要な権限

- **Microsoft.Authorization/roleAssignments/delete**

例7.50 コンピューティングリソースを削除するために必要な権限

- **Microsoft.Compute/disks/delete**
- **Microsoft.Compute/galleries/delete**
- **Microsoft.Compute/galleries/images/delete**
- **Microsoft.Compute/galleries/images/versions/delete**
- **Microsoft.Compute/virtualMachines/delete**
- **Microsoft.Compute/images/delete**

例7.51 Required permissions for deleting identity management resources

- **Microsoft.ManagedIdentity/userAssignedIdentities/delete**

例7.52 ネットワークリソースを削除するために必要な権限

- **Microsoft.Network/dnszones/read**
- **Microsoft.Network/dnsZones/A/read**
- **Microsoft.Network/dnsZones/A/delete**
- **Microsoft.Network/dnsZones/CNAME/read**
- **Microsoft.Network/dnsZones/CNAME/delete**
- **Microsoft.Network/loadBalancers/delete**

- **Microsoft.Network/networkInterfaces/delete**
- **Microsoft.Network/networkSecurityGroups/delete**
- **Microsoft.Network/privateDnsZones/read**
- **Microsoft.Network/privateDnsZones/A/read**
- **Microsoft.Network/privateDnsZones/delete**
- **Microsoft.Network/privateDnsZones/virtualNetworkLinks/delete**
- **Microsoft.Network/publicIPAddresses/delete**
- **Microsoft.Network/virtualNetworks/delete**

例7.53 リソースの正常性をチェックするために必要なアクセス許可

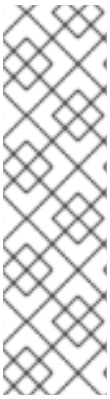
- **Microsoft.Resourcehealth/healthevent/Activated/action**
- **Microsoft.Resourcehealth/healthevent/Resolved/action**
- **Microsoft.Resourcehealth/healthevent/Updated/action**

例7.54 リソースグループを削除するために必要なアクセス許可

- **Microsoft.Resources/subscriptions/resourcegroups/delete**

例7.55 ストレージリソースを削除するために必要な権限

- **Microsoft.Storage/storageAccounts/delete**
- **Microsoft.Storage/storageAccounts/listKeys/action**



注記

Azure に OpenShift Container Platform をインストールするには、リソースグループの作成に関連するアクセス許可をサブスクリプションに限定する必要があります。リソースグループが作成されたら、作成されたリソースグループに残りのアクセス許可のスコープを設定できます。パブリック DNS ゾーンが別のリソースグループに存在する場合は、ネットワーク DNS ゾーンに関連するアクセス許可を常にサブスクリプションに適用する必要があります。

OpenShift Container Platform クラスターを削除するときに、すべてのパーミッションをサブスクリプションに限定できます。

7.10.2.7. サービスプリンシパルの作成

OpenShift Container Platform とそのインストールプログラムは Azure Resource Manager を使用して Microsoft Azure リソースを作成するため、それを表すサービスプリンシパルを作成する必要があります。

前提条件

- Azure CLI のインストールまたは更新を実行します。
- Azure アカウントには、使用するサブスクリプションに必要なロールがなければなりません。
- カスタムロールを使用する場合は、**ユーザーによってプロビジョニングされたインフラストラクチャーに必要な Azure アクセス許可** セクションに記載されている必要なアクセス許可を持つカスタムロール を作成しておきます。

手順

1. Azure CLI にログインします。

```
$ az login
```

2. Azure アカウントでサブスクリプションを使用している場合は、適切なサブスクリプションを使用していることを確認してください。
   a. 利用可能なアカウントの一覧を表示し、クラスターに使用するサブスクリプションの **tenantId** の値を記録します。

```
$ az account list --refresh
```

出力例

```
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

   b. アクティブなアカウントの詳細を表示し、**tenantId** 値が使用するサブスクリプションと一致することを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
```

```
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** パラメーターの値が正しいサブスクリプション ID であることを確認してください。

c. 適切なサブスクリプションを使用していない場合には、アクティブなサブスクリプションを変更します。

```
$ az account set -s <subscription_id> (1)
```

(1) サブスクリプション ID を指定します。

d. サブスクリプション ID の更新を確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 出力から **tenantId** および **id** パラメーター値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

4. アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role <role_name> \ (1)
    --name <service_principal> \ (2)
    --scopes /subscriptions/<subscription_id> (3)
```

(1) ロール名を定義します。**Contributor** ロールを使用するか、必要なアクセス許可を含むカスタムロールを指定できます。

(2) サービスプリンシパル名を定義します。

(3) サブスクリプション ID を指定します。

出力例

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "displayName": <service_principal>",
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee"
}
```

5. 直前の出力の **appId** および **password** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

6. **Contributor** ロールをサービスプリンシパルに適用した場合は、次のコマンドを実行して **User Administrator Access** ロールを割り当てます。

```
$ az role assignment create --role "User Access Administrator" \
  --assignee-object-id $(az ad sp show --id <appId> --query id -o tsv) 1
```

1 サービスプリンシパルの **appId** パラメーター値を指定します。

関連情報

- CCO モードの詳細は、Cloud Crednetial Operator について を参照してください。

7.10.2.8. サポート対象の Azure リージョン

インストールプログラムは、サブスクリプションに基づいて利用可能な Microsoft Azure リージョンのリストを動的に生成します。

サポート対象の Azure パブリックリージョン

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)
- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)
- **canadaeast** (Canada East)
- **centralindia** (Central India)
- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)

- **eastus2** (East US 2)
- **francecentral** (France Central)
- **germanywestcentral** (Germany West Central)
- **israelcentral** (イスラエル中央)
- **italynorth** (イタリア北部)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **mexicocentral** (Mexico Central)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)
- **norwayeast** (Norway East)
- **polandcentral** (ポーランド中央)
- **qarcentral** (カタール中部)
- **southafricanorth** (South Africa North)
- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)
- **swedencentral** (スウェーデン中央)
- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)
- **uksouth** (UK South)
- **ukwest** (UK West)
- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)

- **westus3**(West US 3)

サポート対象の Azure Government リージョン
以下の Microsoft Azure Government (MAG) リージョンのサポートが OpenShift Container Platform バージョン 4.6 に追加されています。

- **usgovtexas** (US Gov Texas)
- **usgovvirginia** (US Gov Virginia)

Azure ドキュメント の利用可能なすべての MAG リージョンを参照できます。他の提供される MAG リージョンは OpenShift Container Platform で機能することが予想されますが、まだテストされていません。

7.10.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

7.10.3.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表7.20 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選

択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

7.10.3.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表7.21 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

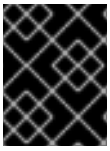
3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。



重要

premiumIO パラメーターが **true** に設定されている Azure 仮想マシンを使用する必要があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

7.10.3.3. Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure インスタンスタイプは OpenShift Container Platform でテストされています。

例7.56 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***

- **t3.***
- **t3a.***

7.10.3.4. 64 ビット ARM インフラストラクチャー上の Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure Azure64 インスタンスタイプは OpenShift Container Platform でテストされています。

例7.57 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

7.10.4. Azure Marketplace オファリングの使用

Azure Marketplace を使用すると、OpenShift Container Platform クラスターをデプロイできます。これは、Azure を通じて従量課金制 (時間単位、コア単位) で請求され、Red Hat の直接サポートも受けることができます。

Azure Marketplace オファリングを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に Azure Marketplace イメージを取得する必要があります。インストールプログラムは、このイメージを使用してワーカーまたはコントロールプレーンノードをデプロイします。イメージを取得するときは、次の点を考慮してください。

- イメージは同じですが、Azure Marketplace のパブリシャーは地域によって異なります。北米にお住まいの場合は、**redhat** をパブリッシャーとして指定してください。EMEA にお住まいの場合は、**redhat-limited** をパブリッシャーとして指定してください。
- このオファーには、**rh-ocp-worker** SKU と **rh-ocp-worker-gen1** SKU が含まれています。**rh-ocp-worker** SKU は、Hyper-V 世代のバージョン 2 VM イメージを表します。OpenShift Container Platform で使用されるデフォルトのインスタンスタイプは、バージョン 2 と互換性があります。バージョン 1 のみと互換性のあるインスタンスタイプを使用する場合は、**rh-ocp-worker-gen1** SKU に関連付けられたイメージを使用します。**rh-ocp-worker-gen1** SKU は、Hyper-V バージョン 1 VM イメージを表します。

重要

Azure マーケットプレイスを使用したイメージのインストールは、64 ビット ARM インスタンスを備えたクラスターではサポートされていません。

前提条件

- Azure CLI クライアント **(az)** をインストールしている。
- お客様の Azure アカウントにはオファーのエンタイトルメントがあり、Azure CLI クライアントを使用してこのアカウントにログインしている。

手順

1. 以下のいずれかのコマンドを実行して、利用可能なすべての OpenShift Container Platform イメージを表示します。

   - 北米:

     ```
     $  az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
     ```

     出力例

     ```
     Offer          Publisher       Sku                 Urn                                                                  Version
     -------------  --------------  ------------------  -------------------------------------------------------------------  -----------------
     rh-ocp-worker  RedHat          rh-ocp-worker       RedHat:rh-ocp-worker:rh-ocp-worker:413.92.2023101700             413.92.2023101700
     rh-ocp-worker  RedHat          rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-gen1:413.92.2023101700         413.92.2023101700
     ```

   - EMEA:

     ```
     $  az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
     ```

     出力例

     ```
     Offer          Publisher       Sku                 Urn                                                                  Version
     -------------  --------------  ------------------  -------------------------------------------------------------------  -----------------
     rh-ocp-worker  redhat-limited  rh-ocp-worker       redhat-limited:rh-ocp-worker:rh-ocp-worker:413.92.2023101700             413.92.2023101700
     rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-worker-gen1:413.92.2023101700         413.92.2023101700
     ```

     注記

     コンピュートおよびコントロールプレーンノードで利用可能な最新のイメージを使用します。必要に応じて、VM はインストールプロセスの一部として自動的にアップグレードされます。

2. 次のいずれかのコマンドを実行して、オファーのイメージを調べます。

   - 北米:

     ```
     $ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - EMEA:

     ```
     $ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

3. 次のコマンドのいずれかを実行して、オファーの条件を確認します。

   - 北米:

     ```
     $ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

- EMEA:

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. 次のコマンドのいずれかを実行して、オファリングの条件に同意します。

- 北米:

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. オファーのイメージの詳細を記録します。Azure Resource Manager (ARM)テンプレートを使用してコンピュートノードをデプロイする場合：

a. **id** パラメーターを削除し、オファーの値を使用して、**offer**、**publisher**、**sku**、および **version** パラメーターを追加して、**storageProfile.imageReference** を更新します。

b. 仮想マシン (VM) の **plan** を指定します。

更新された storageProfile.imageReference オブジェクトと指定された plan を含む 06_workers.json ARM テンプレートの例

```
...
  "plan" : {
    "name": "rh-ocp-worker",
    "product": "rh-ocp-worker",
    "publisher": "redhat"
  },
  "dependsOn" : [
    "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')
[copyIndex()], '-nic'))]"
  ],
  "properties" : {
...
  "storageProfile": {
    "imageReference": {
    "offer": "rh-ocp-worker",
    "publisher": "redhat",
    "sku": "rh-ocp-worker",
    "version": "413.92.2023101700"
    }
    ...
  }
...
  }
```

7.10.4.1. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

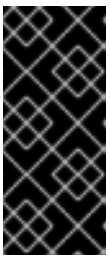
前提条件

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.10.4.2. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバックまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

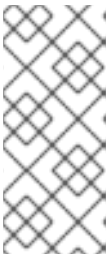
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

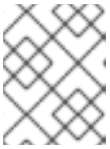
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

7.10.5. Azure のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Microsoft Azure にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

7.10.5.1. オプション: 別個の /var パーティションの作成

OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

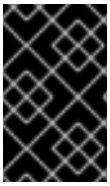
OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。

- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。



重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、**manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

7.10.5.2. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、アクセス可能な場所にアップロードしました。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルを使用してクラスターをインストールしている場合は、そのアプリケーション ID とパスワードが必要です。
- システムが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、インストールプログラムを実行する仮想マシン上でそれが有効になっています。
- ユーザーが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、次の前提条件を満たしている必要があります。
  - そのクライアント ID がある。
  - これは、インストールプログラムを実行する仮想マシンに割り当てられている。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルまたはマネージド ID を使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
   このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。
2. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。
インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。

iii. サブスクリプションの次の Azure パラメーター値を入力します。

- **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
- **azure tenant id** テナント ID を入力します。

iv. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアント ID** の入力を求められたら、次のいずれかを行います。

- サービスプリンシパルを使用している場合は、そのアプリケーション ID を入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、そのクライアント ID を指定します。

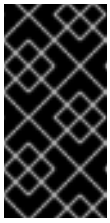
v. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアントシークレット** の入力を求められたら、次のいずれかを実行します。

- サービスプリンシパルを使用している場合は、そのパスワードを入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

vi. クラスターをデプロイするリージョンを選択します。

vii. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

viii. クラスターの記述名を入力します。



重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

ix. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. **platform.azure** フィールドでクラスターをインストールするための VNet のネットワークとサブネットを定義します。

```
networkResourceGroupName: <vnet_resource_group> 1
virtualNetwork: <vnet> 2
controlPlaneSubnet: <control_plane_subnet> 3
computeSubnet: <compute_subnet> 4
```

1. **<vnet_resource_group>** を、既存の仮想ネットワーク (VNet) を含むリソースグループ名に置き換えます。
2. **<vnet>** を既存の仮想ネットワーク名に置き換えます。
3. **<control_plane_subnet>** は、コントロールプレーンマシンをデプロイする既存のサブネット名に置き換えます。
4. **<compute_subnet>** を、コンピュートマシンをデプロイするための既存のサブネット名に置き換えます。

d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

e. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

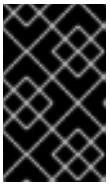
このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

重要

Azure Firewall は、Azure Public ロードバランサーと シームレスに連携しません。したがって、インターネットアクセスを制限するために Azure Firewall を使用する場合は、**install-config.yaml** の **publish** フィールドを **Internal** に設定する必要があります。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの **~/.azure/** ディレクトリーに保存します。これにより、インストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作

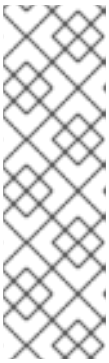
成するときにプロファイルをロードできるようになります。

7.10.5.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

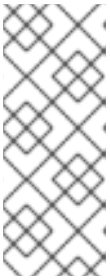
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

7.10.5.4. ARM テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Microsoft Azure で実行するのに役立つ指定の Azure Resource Manager (ARM) テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。

注記

特定の ARM テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. 提供される ARM テンプレートで使用される **install-config.yaml** にある一般的な変数をエクスポートします。

```
$ export CLUSTER_NAME=<cluster_name> (1)
$ export AZURE_REGION=<azure_region> (2)
$ export SSH_KEY=<ssh_key> (3)
$ export BASE_DOMAIN=<base_domain> (4)
$ export BASE_DOMAIN_RESOURCE_GROUP=<base_domain_resource_group> (5)
```

(1) **install-config.yaml** ファイルからの **.metadata.name** 属性の値。

(2) クラスターをデプロイするリージョン (例: **centralus**)。これは、**install-config.yaml** ファイルからの **.platform.azure.region** 属性の値です。

(3) 文字列としての SSH RSA 公開鍵ファイル。SSH キーは、スペースが含まれているために引用符で囲む必要があります。これは、**install-config.yaml** ファイルからの **.sshKey** 属性の値です。

(4) クラスターをデプロイするベースドメイン。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。これは、**install-config.yaml** からの **.baseDomain** 属性の値です。

(5) パブリック DNS ゾーンが存在するリソースグループ。これは、**install-config.yaml** ファイルからの **.platform.azure.baseDomainResourceGroupName** 属性の値です。

以下に例を示します。

```
$ export CLUSTER_NAME=test-cluster
$ export AZURE_REGION=centralus
$ export SSH_KEY="ssh-rsa xxx/xxx/xxx= user@email.com"
$ export BASE_DOMAIN=example.com
$ export BASE_DOMAIN_RESOURCE_GROUP=ocp-cluster
```

2. kubeadmin 認証情報をエクスポートします。

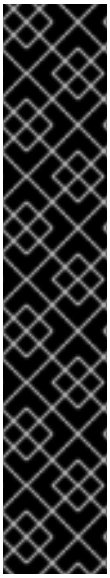
```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
```

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

7.10.5.5. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

4. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

重要

ユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールするときに **MachineAPI** 機能を無効にした場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除する必要があります。そうしないと、クラスターのインストールに失敗します。

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. ユーザーによってプロビジョニングされるインフラストラクチャーで Azure を設定する場合、Azure Resource Manager (ARM) テンプレートで後に使用するためにマニフェストファイルに定義された一般的な変数の一部をエクスポートする必要があります。

   a. 以下のコマンドを使用してインフラストラクチャー ID をエクスポートします。

```
$ export INFRA_ID=<infra_id> 1
```

1 OpenShift Container Platform クラスターには、**<cluster_name>-<random_string>** の形式の識別子 (**INFRA_ID**) が割り当てられます。これは、提供される ARM テンプレートを使用して作成されるほとんどのリソースのベース名として使用されます。こ

れは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.infrastructureName** 属性の値です。

b. 以下のコマンドを使用してリソースグループをエクスポートします。

```
$ export RESOURCE_GROUP=<resource_group> 1
```

1 この Azure デプロイメントで作成されたすべてのリソースは、リソースグループ の一部として存在します。リソースグループ名は、**<cluster_name>-<random_string>-rg** 形式の **INFRA_ID** をベースとしています。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.platformStatus.azure.resourceGroupName** 属性の値です。

8. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

7.10.6. Azure リソースグループの作成

Microsoft Azure リソースグループ およびリソースグループのアイデンティティーを作成する必要があります。これらはいずれも Azure での OpenShift Container Platform クラスターのインストール時に使用されます。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. サポートされる Azure リージョンにリソースグループを作成します。

```
$ az group create --name ${RESOURCE_GROUP} --location ${AZURE_REGION}
```

2. リソースグループの Azure アイデンティティーを作成します。

```
$ az identity create -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity
```

これは、クラスター内の Operator に必要なアクセスを付与するために使用されます。たとえば、これにより Ingress Operator はパブリック IP およびそのロードバランサーを作成できます。Azure アイデンティティーをロールに割り当てる必要があります。

3. Contributor ロールを Azure アイデンティティーに付与します。

a. Azure ロールの割り当てで必要な以下の変数をエクスポートします。

```
$ export PRINCIPAL_ID=`az identity show -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity --query principalId --out tsv`
```

```
$ export RESOURCE_GROUP_ID=`az group show -g ${RESOURCE_GROUP} --query id --out tsv`
```

b. Contributor ロールをアイデンティティーに割り当てます。

```
$ az role assignment create --assignee "${PRINCIPAL_ID}" --role 'Contributor' --scope "${RESOURCE_GROUP_ID}"
```

注記

必要なすべてのアクセス許可を持つカスタムロールを ID に割り当てる場合は、次のコマンドを実行します。

```
$ az role assignment create --assignee "${PRINCIPAL_ID}" --role <custom_role> \ 1
--scope "${RESOURCE_GROUP_ID}"
```

1 カスタムロール名を指定します。

7.10.7. RHCOS クラスターイメージおよびブートストラップ Ignition 設定ファイルのアップロード

Azure クライアントは、ローカルに存在するファイルに基づくデプロイメントをサポートしていません。RHCOS 仮想ハードディスク (VHD) クラスターイメージとブートストラップ Ignition 設定ファイルをコピーしてストレージコンテナーに保存し、デプロイメント中にアクセスできるようにする必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. VHD クラスターイメージを保存するために Azure ストレージアカウントを作成します。

```
$ az storage account create -g ${RESOURCE_GROUP} --location ${AZURE_REGION} --
name ${CLUSTER_NAME}sa --kind Storage --sku Standard_LRS
```

警告

Azure ストレージアカウント名は 3 文字から 24 文字の長さで、数字および小文字のみを使用する必要があります。**CLUSTER_NAME** 変数がこれらの制限に準拠しない場合、Azure ストレージアカウント名を手動で定義する必要があります。Azure ストレージアカウント名の制限についての詳細は、Azure ドキュメントの Resolve errors for storage account names を参照してください。

2. ストレージアカウントキーを環境変数としてエクスポートします。

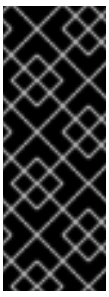
```
$ export ACCOUNT_KEY=`az storage account keys list -g ${RESOURCE_GROUP} --
account-name ${CLUSTER_NAME}sa --query "[0].value" -o tsv`
```

3. RHCOS VHD の URL を環境変数にエクスポートします。

```
$ export VHD_URL=`openshift-install coreos print-stream-json | jq -r '.architectures.
<architecture>."rhel-coreos-extensions"."azure-disk".url'`
```

ここでは、以下のようになります。

<architecture>
アーキテクチャーを指定します。有効な値は **x86_64** または **aarch64** です。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージを指定する必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

4. VHD のストレージコンテナーを作成します。

```
$ az storage container create --name vhd --account-name ${CLUSTER_NAME}sa --account-
key ${ACCOUNT_KEY}
```

5. ローカル VHD を blob にコピーします。

```
$ az storage blob copy start --account-name ${CLUSTER_NAME}sa --account-key
${ACCOUNT_KEY} --destination-blob "rhcos.vhd" --destination-container vhd --source-uri
"${VHD_URL}"
```

6. blob ストレージコンテナーを作成し、生成された **bootstrap.ign** ファイルをアップロードします。

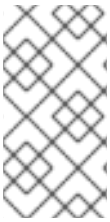
```
$ az storage container create --name files --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY}
```

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key
${ACCOUNT_KEY} -c "files" -f "<installation_directory>/bootstrap.ign" -n "bootstrap.ign"
```

7.10.8. DNS ゾーンの作成例

DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターに必要です。シナリオに適した DNS ストラテジーを選択する必要があります。

この例の場合、Azure の DNS ソリューション が使用されるため、外部 (インターネット) の可視性のために新規パブリック DNS ゾーンと、内部クラスターの解決用にプライベート DNS ゾーンが作成されます。



注記

パブリック DNS ゾーンは、クラスターデプロイメントと同じリソースグループに存在している必要はなく、必要なベースドメイン用にすでに組織内に存在している可能性があります。その場合、パブリック DNS ゾーンの作成を省略できます。先に生成したインストール設定がこのシナリオに基づいていることを確認してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. **BASE_DOMAIN_RESOURCE_GROUP** 環境変数でエクスポートされたリソースグループに、新規のパブリック DNS ゾーンを作成します。

```
$ az network dns zone create -g ${BASE_DOMAIN_RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

   すでに存在するパブリック DNS ゾーンを使用している場合は、この手順を省略できます。

2. このデプロイメントの残りの部分と同じリソースグループにプライベート DNS ゾーンを作成します。

```
$ az network private-dns zone create -g ${RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

Azure でのパブリック DNS ゾーンの設定 についてのセクションを参照してください。

7.10.9. Azure での VNet の作成

OpenShift Container Platform クラスター用に Microsoft Azure で使用する仮想ネットワーク (VNet) を作成する必要があります。各種の要件を満たすように VPC をカスタマイズできます。VNet を作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

提供される ARM テンプレートを使用して Azure インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VNet の ARM テンプレート**セクションからテンプレートをコピーし、これを **01_vnet.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要な VNet について記述しています。
2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/01_vnet.json" \
  --parameters baseName="${INFRA_ID}" 1
```

1 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. VNet テンプレートをプライベート DNS ゾーンにリンクします。

```
$ az network private-dns link vnet create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n ${INFRA_ID}-network-link -v "${INFRA_ID}-vnet"
-e false
```

7.10.9.1. VNet の ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な VNet をデプロイすることができます。

例7.58 01_vnet.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
```

```
    "metadata" : {
      "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
    }
   }
  },
  "variables" : {
   "location" : "[resourceGroup().location]",
   "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
   "addressPrefix" : "10.0.0.0/16",
   "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
   "masterSubnetPrefix" : "10.0.0.0/24",
   "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
   "nodeSubnetPrefix" : "10.0.1.0/24",
   "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]"
  },
  "resources" : [
   {
    "apiVersion" : "2018-12-01",
    "type" : "Microsoft.Network/virtualNetworks",
    "name" : "[variables('virtualNetworkName')]",
    "location" : "[variables('location')]",
    "dependsOn" : [
     "[concat('Microsoft.Network/networkSecurityGroups/', variables('clusterNsgName'))]"
    ],
    "properties" : {
     "addressSpace" : {
      "addressPrefixes" : [
       "[variables('addressPrefix')]"
      ]
     },
     "subnets" : [
      {
       "name" : "[variables('masterSubnetName')]",
       "properties" : {
        "addressPrefix" : "[variables('masterSubnetPrefix')]",
        "serviceEndpoints": [],
        "networkSecurityGroup" : {
         "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
        }
       }
      },
      {
       "name" : "[variables('nodeSubnetName')]",
       "properties" : {
        "addressPrefix" : "[variables('nodeSubnetPrefix')]",
        "serviceEndpoints": [],
        "networkSecurityGroup" : {
         "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
        }
       }
      }
     ]
    }
   },
```

```
    {
      "type" : "Microsoft.Network/networkSecurityGroups",
      "name" : "[variables('clusterNsgName')]",
      "apiVersion" : "2018-10-01",
      "location" : "[variables('location')]",
      "properties" : {
        "securityRules" : [
          {
            "name" : "apiserver_in",
            "properties" : {
              "protocol" : "Tcp",
              "sourcePortRange" : "*",
              "destinationPortRange" : "6443",
              "sourceAddressPrefix" : "*",
              "destinationAddressPrefix" : "*",
              "access" : "Allow",
              "priority" : 101,
              "direction" : "Inbound"
            }
          }
        ]
      }
    }
  ]
}
```

7.10.10. Azure インフラストラクチャー用の RHCOS クラスターイメージのデプロイ

OpenShift Container Platform ノードに Microsoft Azure 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- RHCOS 仮想ハードディスク (VHD) クラスターイメージを Azure ストレージコンテナーに保存します。
- ブートストラップ Ignition 設定ファイルを Azure ストレージコンテナーに保存します。

手順

1. 本トピックの **イメージストレージの ARM テンプレート**セクションからテンプレートをコピーし、これを **02_storage.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なイメージストレージ について記述しています。
2. RHCOS VHD blob URL を変数としてエクスポートします。

```
$ export VHD_BLOB_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -o tsv`
```

3. クラスターイメージのデプロイ

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/02_storage.json" \
  --parameters vhdBlobURL="${VHD_BLOB_URL}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameters storageAccount="${CLUSTER_NAME}sa" \ 3
  --parameters architecture="<architecture>" 4
```

1. マスターマシンおよびワーカーマシンを作成するために使用される RHCOS VHD の blob URL。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. Azure ストレージアカウントの名前。
4. システムアーキテクチャーを指定します。有効な値は、**x64** (デフォルト) または **Arm64** です。

7.10.10.1. イメージストレージの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な保存された Red Hat Enterprise Linux CoreOS (RHCOS) をデプロイすることができます。

例7.59 02_storage.json ARM テンプレート

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "architecture": {
      "type": "string",
      "metadata": {
        "description": "The architecture of the Virtual Machines"
      },
      "defaultValue": "x64",
      "allowedValues": [
        "Arm64",
        "x64"
      ]
    },
    "baseName": {
      "type": "string",
      "minLength": 1,
      "metadata": {
        "description": "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "storageAccount": {
      "type": "string",
      "metadata": {
        "description": "The Storage Account name"
      }
    },
```

```
    "vhdBlobURL": {
      "type": "string",
      "metadata": {
        "description": "URL pointing to the blob where the VHD to be used to create master and
worker machines is located"
      }
    }
  },
  "variables": {
    "location": "[resourceGroup().location]",
    "galleryName": "[concat('gallery_', replace(parameters('baseName'), '-', '_'))]",
    "imageName": "[parameters('baseName')]",
    "imageNameGen2": "[concat(parameters('baseName'), '-gen2')]",
    "imageRelease": "1.0.0"
  },
  "resources": [
    {
      "apiVersion": "2021-10-01",
      "type": "Microsoft.Compute/galleries",
      "name": "[variables('galleryName')]",
      "location": "[variables('location')]",
      "resources": [
        {
          "apiVersion": "2021-10-01",
          "type": "images",
          "name": "[variables('imageName')]",
          "location": "[variables('location')]",
          "dependsOn": [
            "[variables('galleryName')]"
          ],
          "properties": {
            "architecture": "[parameters('architecture')]",
            "hyperVGeneration": "V1",
            "identifier": {
              "offer": "rhcos",
              "publisher": "RedHat",
              "sku": "basic"
            },
            "osState": "Generalized",
            "osType": "Linux"
          },
          "resources": [
            {
              "apiVersion": "2021-10-01",
              "type": "versions",
              "name": "[variables('imageRelease')]",
              "location": "[variables('location')]",
              "dependsOn": [
                "[variables('imageName')]"
              ],
              "properties": {
                "publishingProfile": {
                  "storageAccountType": "Standard_LRS",
                  "targetRegions": [
                    {
                      "name": "[variables('location')]",
```

```
                "regionalReplicaCount": "1"
              }
            ]
          },
          "storageProfile": {
            "osDiskImage": {
              "source": {
                "id": "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccount'))]",
                "uri": "[parameters('vhdBlobURL')]"
              }
            }
          }
        }
      ]
    },
    {
      "apiVersion": "2021-10-01",
      "type": "images",
      "name": "[variables('imageNameGen2')]",
      "location": "[variables('location')]",
      "dependsOn": [
        "[variables('galleryName')]"
      ],
      "properties": {
        "architecture": "[parameters('architecture')]",
        "hyperVGeneration": "V2",
        "identifier": {
          "offer": "rhcos-gen2",
          "publisher": "RedHat-gen2",
          "sku": "gen2"
        },
        "osState": "Generalized",
        "osType": "Linux"
      },
      "resources": [
        {
          "apiVersion": "2021-10-01",
          "type": "versions",
          "name": "[variables('imageRelease')]",
          "location": "[variables('location')]",
          "dependsOn": [
            "[variables('imageNameGen2')]"
          ],
          "properties": {
            "publishingProfile": {
              "storageAccountType": "Standard_LRS",
              "targetRegions": [
                {
                  "name": "[variables('location')]",
                  "regionalReplicaCount": "1"
                }
              ]
            },
            "storageProfile": {
```

```
                "osDiskImage": {
                  "source": {
                    "id": "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccount'))]",
                    "uri": "[parameters('vhdBlobURL')]"
                  }
                }
              }
            }
          }
        ]
      }
    ]
  }
}
```

7.10.11. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

7.10.11.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表7.22 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート

プロトコル	ポート	説明
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表7.23 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表7.24 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

7.10.12. Azure でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用するネットワークおよび負荷分散を Microsoft Azure で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

提供される ARM テンプレートを使用して Azure インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。

手順

1. 本トピックの **ネットワークおよびロードばランサーの ARM テンプレート**セクションからテンプレートをコピーし、これを **03_infra.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。
2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/03_infra.json" \
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

1. プライベート DNS ゾーンの名前。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. API パブリックロードバランサーのパブリックゾーンに **api** DNS レコードを作成します。**${BASE_DOMAIN_RESOURCE_GROUP}** 変数は、パブリック DNS ゾーンがあるリソースグループをポイントする必要があります。
   a. 以下の変数をエクスポートします。

```
$ export PUBLIC_IP=`az network public-ip list -g ${RESOURCE_GROUP} --query "[?
name=='${INFRA_ID}-master-pip'] | [0].ipAddress" -o tsv`
```

   b. 新しいパブリックゾーンに **api** DNS レコードを作成します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n api -a ${PUBLIC_IP} --ttl 60
```

   クラスターを既存のパブリックゾーンに追加する場合は、**api** DNS レコードを代わりに作成できます。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n api.${CLUSTER_NAME} -a ${PUBLIC_IP} --ttl 60
```

7.10.12.1. ネットワークおよびロードバランサーの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用して、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例7.60 03_infra.json ARM テンプレート

```
{
```

```
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vnetBaseName": {
      "type": "string",
      "defaultValue": "",
      "metadata" : {
        "description" : "The specific customer vnet's base name (optional)"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterPublicIpAddressName" : "[concat(parameters('baseName'), '-master-pip')]",
    "masterPublicIpAddressID" : "[resourceId('Microsoft.Network/publicIPAddresses',
variables('masterPublicIpAddressName'))]",
    "masterLoadBalancerName" : "[parameters('baseName')]",
    "masterLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('masterLoadBalancerName'))]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "internalLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('internalLoadBalancerName'))]",
    "skuName": "Standard"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('masterPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "properties" : {
```

```
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('masterPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/loadBalancers",
      "name" : "[variables('masterLoadBalancerName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/publicIPAddresses/', variables('masterPublicIpAddressName'))]"
      ],
      "properties" : {
        "frontendIPConfigurations" : [
          {
            "name" : "public-lb-ip-v4",
            "properties" : {
              "publicIPAddress" : {
                "id" : "[variables('masterPublicIpAddressID')]"
              }
            }
          }
        ],
        "backendAddressPools" : [
          {
            "name" : "[variables('masterLoadBalancerName')]"
          }
        ],
        "loadBalancingRules" : [
          {
            "name" : "api-internal",
            "properties" : {
              "frontendIPConfiguration" : {
                "id" :"[concat(variables('masterLoadBalancerID'), '/frontendIPConfigurations/public-lb-ip-
v4')]"
              },
              "backendAddressPool" : {
                "id" : "[concat(variables('masterLoadBalancerID'), '/backendAddressPools/',
variables('masterLoadBalancerName'))]"
              },
              "protocol" : "Tcp",
              "loadDistribution" : "Default",
              "idleTimeoutInMinutes" : 30,
              "frontendPort" : 6443,
              "backendPort" : 6443,
              "probe" : {
                "id" : "[concat(variables('masterLoadBalancerID'), '/probes/api-internal-probe')]"
              }
            }
          }
        ],
```

```
        "probes" : [
          {
            "name" : "api-internal-probe",
            "properties" : {
              "protocol" : "Https",
              "port" : 6443,
              "requestPath": "/readyz",
              "intervalInSeconds" : 10,
              "numberOfProbes" : 3
            }
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/loadBalancers",
      "name" : "[variables('internalLoadBalancerName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "properties" : {
        "frontendIPConfigurations" : [
          {
            "name" : "internal-lb-ip",
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "subnet" : {
                "id" : "[variables('masterSubnetRef')]"
              },
              "privateIPAddressVersion" : "IPv4"
            }
          }
        ],
        "backendAddressPools" : [
          {
            "name" : "internal-lb-backend"
          }
        ],
        "loadBalancingRules" : [
          {
            "name" : "api-internal",
            "properties" : {
              "frontendIPConfiguration" : {
                "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
              },
              "frontendPort" : 6443,
              "backendPort" : 6443,
              "enableFloatingIP" : false,
              "idleTimeoutInMinutes" : 30,
              "protocol" : "Tcp",
              "enableTcpReset" : false,
              "loadDistribution" : "Default",
              "backendAddressPool" : {
```

```
            "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-backend')]"
           },
           "probe" : {
            "id" : "[concat(variables('internalLoadBalancerID'), '/probes/api-internal-probe')]"
           }
          }
         },
         {
          "name" : "sint",
          "properties" : {
           "frontendIPConfiguration" : {
            "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-ip')]"
           },
           "frontendPort" : 22623,
           "backendPort" : 22623,
           "enableFloatingIP" : false,
           "idleTimeoutInMinutes" : 30,
           "protocol" : "Tcp",
           "enableTcpReset" : false,
           "loadDistribution" : "Default",
           "backendAddressPool" : {
            "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-backend')]"
           },
           "probe" : {
            "id" : "[concat(variables('internalLoadBalancerID'), '/probes/sint-probe')]"
           }
          }
         }
        ],
        "probes" : [
         {
          "name" : "api-internal-probe",
          "properties" : {
           "protocol" : "Https",
           "port" : 6443,
           "requestPath": "/readyz",
           "intervalInSeconds" : 10,
           "numberOfProbes" : 3
          }
         },
         {
          "name" : "sint-probe",
          "properties" : {
           "protocol" : "Https",
           "port" : 22623,
           "requestPath": "/healthz",
           "intervalInSeconds" : 10,
           "numberOfProbes" : 3
          }
         }
        ]
       }
    },
```

```
    {
      "apiVersion": "2018-09-01",
      "type": "Microsoft.Network/privateDnsZones/A",
      "name": "[concat(parameters('privateDNSZoneName'), '/api')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
      ],
      "properties": {
        "ttl": 60,
        "aRecords": [
          {
            "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
          }
        ]
      }
    },
    {
      "apiVersion": "2018-09-01",
      "type": "Microsoft.Network/privateDnsZones/A",
      "name": "[concat(parameters('privateDNSZoneName'), '/api-int')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
      ],
      "properties": {
        "ttl": 60,
        "aRecords": [
          {
            "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
          }
        ]
      }
    }
  ]
}
```

7.10.13. Azure でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Microsoft Azure で作成する必要があります。このマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

提供されている ARM テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。

手順

1. 本トピックの **ブートストラップマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. ブートストラップ URL 変数をエクスポートします。

```
$ bootstrap_url_expiry=`date -u -d "10 hours" '+%Y-%m-%dT%H:%MZ'`
```

```
$ export BOOTSTRAP_URL=`az storage blob generate-sas -c 'files' -n 'bootstrap.ign' --https-only --full-uri --permissions r --expiry $bootstrap_url_expiry --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -o tsv`
```

3. ブートストラップ Ignition 変数をエクスポートします。

```
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "3.2.0" --arg url ${BOOTSTRAP_URL} '{ignition:{version:$v,config:{replace:{source:$url}}}}' | base64 | tr -d '\n'`
```

4. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/04_bootstrap.json" \
  --parameters bootstrapIgnition="${BOOTSTRAP_IGNITION}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameter bootstrapVMSize="Standard_D4s_v3" 3
```

1. ブートストラップクラスターのブートストラップ Ignition コンテンツ。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. オプション: ブートストラップ仮想マシンのサイズを指定します。指定したアーキテクチャーと互換性のある仮想マシンサイズを使用してください。この値が定義されていない場合は、テンプレートのデフォルト値が設定されます。

7.10.13.1. ブートストラップマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例7.61 04_bootstrap.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vnetBaseName": {
      "type": "string",
      "defaultValue": "",
      "metadata" : {
        "description" : "The specific customer vnet's base name (optional)"
      }
    },
    "bootstrapIgnition" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Bootstrap ignition content for the bootstrap cluster"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "defaultValue" : "Unused",
      "metadata" : {
        "description" : "Unused"
      }
    },
    "bootstrapVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "metadata" : {
        "description" : "The size of the Bootstrap Virtual Machine"
      }
    },
    "hyperVGen": {
      "type": "string",
      "metadata": {
        "description": "VM generation image to use"
      },
      "defaultValue": "V2",
      "allowedValues": [
        "V1",
        "V2"
      ]
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
```

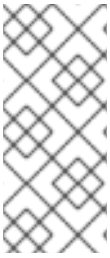
```
    "virtualNetworkName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[parameters('baseName')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "vmName" : "[concat(parameters('baseName'), '-bootstrap')]",
    "nicName" : "[concat(variables('vmName'), '-nic')]",
    "galleryName": "[concat('gallery_', replace(parameters('baseName'), '-', '_'))]",
    "imageName" : "[concat(parameters('baseName'), if(equals(parameters('hyperVGen'), 'V2'), '-
gen2', ''))]",
    "clusterNsgName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-nsg')]",
    "sshPublicIpAddressName" : "[concat(variables('vmName'), '-ssh-pip')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('sshPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('sshPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Network/networkInterfaces",
      "name" : "[variables('nicName')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[resourceId('Microsoft.Network/publicIPAddresses', variables('sshPublicIpAddressName'))]"
      ],
      "properties" : {
        "ipConfigurations" : [
          {
            "name" : "pipConfig",
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('sshPublicIpAddressName'))]"
              },
              "subnet" : {
```

```
            "id" : "[variables('masterSubnetRef')]"
           },
           "loadBalancerBackendAddressPools" : [
            {
             "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/',
variables('masterLoadBalancerName'))]"
            },
            {
             "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
            }
           ]
          }
         }
        ]
       }
      },
      {
       "apiVersion" : "2018-06-01",
       "type" : "Microsoft.Compute/virtualMachines",
       "name" : "[variables('vmName')]",
       "location" : "[variables('location')]",
       "identity" : {
        "type" : "userAssigned",
        "userAssignedIdentities" : {
         "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
       },
       "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
       ],
       "properties" : {
        "hardwareProfile" : {
         "vmSize" : "[parameters('bootstrapVMSize')]"
        },
        "osProfile" : {
         "computerName" : "[variables('vmName')]",
         "adminUsername" : "core",
         "adminPassword" : "NotActuallyApplied!",
         "customData" : "[parameters('bootstrapIgnition')]",
         "linuxConfiguration" : {
          "disablePasswordAuthentication" : false
         }
        },
        "storageProfile" : {
         "imageReference": {
          "id": "[resourceId('Microsoft.Compute/galleries/images', variables('galleryName'),
variables('imageName'))]"
         },
         "osDisk" : {
          "name": "[concat(variables('vmName'),'_OSDisk')]",
          "osType" : "Linux",
```

```
            "createOption" : "FromImage",
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB" : 100
          }
        },
        "networkProfile" : {
          "networkInterfaces" : [
            {
              "id" : "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"
            }
          ]
        }
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type": "Microsoft.Network/networkSecurityGroups/securityRules",
      "name" : "[concat(variables('clusterNsgName'), '/bootstrap_ssh_in')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[resourceId('Microsoft.Compute/virtualMachines', variables('vmName'))]"
      ],
      "properties": {
        "protocol" : "Tcp",
        "sourcePortRange" : "*",
        "destinationPortRange" : "22",
        "sourceAddressPrefix" : "*",
        "destinationAddressPrefix" : "*",
        "access" : "Allow",
        "priority" : 100,
        "direction" : "Inbound"
      }
    }
  ]
}
```

7.10.14. Azure でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Microsoft Azure で作成する必要があります。これらのマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

デフォルトでは、Microsoft Azure はコントロールプレーンマシンとコンピュートマシンを事前設定されたアベイラビリティーゾーンに配置します。コンピュートノードまたはコントロールプレーンノードのアベイラビリティーゾーンを手動で設定できます。これを行うには、仮想マシンリソースの **zones** パラメーターで各可用性ゾーンを指定して、ベンダーの Azure Resource Manager (ARM) テンプレートを変更します。

提供される ARM テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合は、インストールログで Red Hat サポートに接続することを検討してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの **コントロールプレーンマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **05_masters.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. コントロールプレーンマシンのデプロイメントに必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/05_masters.json" \
  --parameters masterIgnition="${MASTER_IGNITION}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameters masterVMSize="Standard_D8s_v3" 3
```

1. コントロールプレーンノードの Ignition コンテンツ。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. オプション: コントロールプレーン仮想マシンのサイズを指定します。指定したアーキテクチャーと互換性のある仮想マシンサイズを使用してください。この値が定義されていない場合は、テンプレートのデフォルト値が設定されます。

7.10.14.1. コントロールプレーンマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例7.62 05_masters.json ARM テンプレート

```
{
```

```
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vnetBaseName": {
      "type": "string",
      "defaultValue": "",
      "metadata" : {
        "description" : "The specific customer vnet's base name (optional)"
      }
    },
    "masterIgnition" : {
      "type" : "string",
      "metadata" : {
        "description" : "Ignition content for the master nodes"
      }
    },
    "numberOfMasters" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 2,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift masters to deploy"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "defaultValue" : "Unused",
      "metadata" : {
        "description" : "Unused"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "defaultValue" : "",
      "metadata" : {
        "description" : "unused"
      }
    },
    "masterVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D8s_v3",
      "metadata" : {
        "description" : "The size of the Master Virtual Machines"
      }
    },
    "diskSizeGB" : {
      "type" : "int",
```

```
      "defaultValue" : 1024,
      "metadata" : {
        "description" : "Size of the Master VM OS disk, in GB"
      }
    },
    "hyperVGen": {
      "type": "string",
      "metadata": {
        "description": "VM generation image to use"
      },
      "defaultValue": "V2",
      "allowedValues": [
        "V1",
        "V2"
      ]
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[parameters('baseName')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "galleryName": "[concat('gallery_', replace(parameters('baseName'), '-', '_'))]",
    "imageName" : "[concat(parameters('baseName'), if(equals(parameters('hyperVGen'), 'V2'), '-
gen2', ''))]",
    "copy" : [
      {
        "name" : "vmNames",
        "count" :  "[parameters('numberOfMasters')]",
        "input" : "[concat(parameters('baseName'), '-master-', copyIndex('vmNames'))]"
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Network/networkInterfaces",
      "copy" : {
        "name" : "nicCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
      "location" : "[variables('location')]",
      "properties" : {
        "ipConfigurations" : [
          {
            "name" : "pipConfig",
```

```
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "subnet" : {
                "id" : "[variables('masterSubnetRef')]"
              },
              "loadBalancerBackendAddressPools" : [
                {
                  "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/',
variables('masterLoadBalancerName'))]"
                },
                {
                  "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
                }
              ]
            }
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Compute/virtualMachines",
      "copy" : {
        "name" : "vmCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name" : "[variables('vmNames')[copyIndex()]]",
      "location" : "[variables('location')]",
      "identity" : {
        "type" : "userAssigned",
        "userAssignedIdentities" : {
          "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]"
      ],
      "properties" : {
        "hardwareProfile" : {
          "vmSize" : "[parameters('masterVMSize')]"
        },
        "osProfile" : {
          "computerName" : "[variables('vmNames')[copyIndex()]]",
          "adminUsername" : "core",
          "adminPassword" : "NotActuallyApplied!",
          "customData" : "[parameters('masterIgnition')]",
          "linuxConfiguration" : {
            "disablePasswordAuthentication" : false
          }
        },
```

```
        "storageProfile" : {
          "imageReference": {
            "id": "[resourceId('Microsoft.Compute/galleries/images', variables('galleryName'),
variables('imageName'))]"
          },
          "osDisk" : {
            "name": "[concat(variables('vmNames')[copyIndex()], '_OSDisk')]",
            "osType" : "Linux",
            "createOption" : "FromImage",
            "caching": "ReadOnly",
            "writeAcceleratorEnabled": false,
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB" : "[parameters('diskSizeGB')]"
          }
        },
        "networkProfile" : {
          "networkInterfaces" : [
            {
              "id" : "[resourceId('Microsoft.Network/networkInterfaces', concat(variables('vmNames')
[copyIndex()], '-nic'))]",
              "properties": {
                "primary": false
              }
            }
          ]
        }
      }
    }
  ]
}
```

7.10.15. ブートストラップの完了を待機し、Azure のブートストラップリソースを削除する

Microsoft Azure ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ az network nsg rule delete -g ${RESOURCE_GROUP} --nsg-name ${INFRA_ID}-nsg --
name bootstrap_ssh_in
$ az vm stop -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm deallocate -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap --yes
$ az disk delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap_OSDisk --no-
wait --yes
$ az network nic delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-nic --no-
wait
$ az storage blob delete --account-key ${ACCOUNT_KEY} --account-name
${CLUSTER_NAME}sa --container-name files --name bootstrap.ign
$ az network public-ip delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-
ssh-pip
```

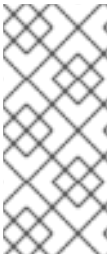
注記

ブートストラップサーバーを削除しないと、API トラフィックがブートストラップサーバーにルーティングされるため、インストールが成功しない場合があります。

7.10.16. Azure での追加のワーカーマシンの作成

Microsoft Azure でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Azure Resource Manager (ARM) テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_workers.json** というタイプのリソースを追加して起動することができます。



注記

デフォルトでは、Microsoft Azure はコントロールプレーンマシンとコンピュートマシンを事前設定されたアベイラビリティーゾーンに配置します。コンピュートノードまたはコントロールプレーンノードのアベイラビリティーゾーンを手動で設定できます。これを行うには、仮想マシンリソースの **zones** パラメーターに各アベイラビリティーゾーンを指定して、ベンダーの ARM テンプレートを変更します。

提供される ARM テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合は、インストールログで Red Hat サポートに接続することを検討してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの **ワーカーマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **06_workers.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. ワーカーマシンのデプロイメントで必要な以下の変数をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/06_workers.json" \
  --parameters workerIgnition="${WORKER_IGNITION}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameters nodeVMSize="Standard_D4s_v3" 3
```

1. ワーカーノードの Ignition コンテンツ。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. オプション: コンピュートノード仮想マシンのサイズを指定します。指定したアーキテクチャーと互換性のある仮想マシンサイズを使用してください。この値が定義されていない場合は、テンプレートのデフォルト値が設定されます。

7.10.16.1. ワーカーマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例7.63 06_workers.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vnetBaseName": {
      "type": "string",
      "defaultValue": "",
      "metadata" : {
        "description" : "The specific customer vnet's base name (optional)"
      }
    },
    "workerIgnition" : {
      "type" : "string",
      "metadata" : {
        "description" : "Ignition content for the worker nodes"
      }
    },
    "numberOfNodes" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 2,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift compute nodes to deploy"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "defaultValue" : "Unused",
      "metadata" : {
        "description" : "Unused"
      }
    },
    "nodeVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "metadata" : {
        "description" : "The size of the each Node Virtual Machine"
      }
    },
    "hyperVGen": {
```

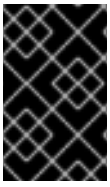
```
      "type": "string",
      "metadata": {
        "description": "VM generation image to use"
      },
      "defaultValue": "V2",
      "allowedValues": [
        "V1",
        "V2"
      ]
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "nodeSubnetName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-worker-subnet')]",
    "nodeSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('nodeSubnetName'))]",
    "infraLoadBalancerName" : "[parameters('baseName')]",
    "sshKeyPath" : "/home/capi/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "galleryName": "[concat('gallery_', replace(parameters('baseName'), '-', '_'))]",
    "imageName" : "[concat(parameters('baseName'), if(equals(parameters('hyperVGen'), 'V2'), '-
gen2', ''))]",
    "copy" : [
      {
        "name" : "vmNames",
        "count" :  "[parameters('numberOfNodes')]",
        "input" : "[concat(parameters('baseName'), '-worker-', variables('location'), '-',
copyIndex('vmNames', 1))]"
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2019-05-01",
      "name" : "[concat('node', copyIndex())]",
      "type" : "Microsoft.Resources/deployments",
      "copy" : {
        "name" : "nodeCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "properties" : {
        "mode" : "Incremental",
        "template" : {
          "$schema" : "http://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
          "contentVersion" : "1.0.0.0",
          "resources" : [
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Network/networkInterfaces",
              "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
```

```
            "location" : "[variables('location')]",
            "properties" : {
              "ipConfigurations" : [
                {
                  "name" : "pipConfig",
                  "properties" : {
                    "privateIPAllocationMethod" : "Dynamic",
                    "subnet" : {
                      "id" : "[variables('nodeSubnetRef')]"
                    }
                  }
                }
              ]
            }
          },
          {
            "apiVersion" : "2018-06-01",
            "type" : "Microsoft.Compute/virtualMachines",
            "name" : "[variables('vmNames')[copyIndex()]]",
            "location" : "[variables('location')]",
            "tags" : {
              "kubernetes.io-cluster-ffranzupi": "owned"
            },
            "identity" : {
              "type" : "userAssigned",
              "userAssignedIdentities" : {
                "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
              }
            },
            "dependsOn" : [
              "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')
[copyIndex()], '-nic'))]"
            ],
            "properties" : {
              "hardwareProfile" : {
                "vmSize" : "[parameters('nodeVMSize')]"
              },
              "osProfile" : {
                "computerName" : "[variables('vmNames')[copyIndex()]]",
                "adminUsername" : "capi",
                "adminPassword" : "NotActuallyApplied!",
                "customData" : "[parameters('workerIgnition')]",
                "linuxConfiguration" : {
                  "disablePasswordAuthentication" : false
                }
              },
              "storageProfile" : {
                "imageReference": {
                  "id": "[resourceId('Microsoft.Compute/galleries/images', variables('galleryName'),
variables('imageName'))]"
                },
                "osDisk" : {
                  "name": "[concat(variables('vmNames')[copyIndex()],'_OSDisk')]",
                  "osType" : "Linux",
                  "createOption" : "FromImage",
```

```
                "managedDisk": {
                  "storageAccountType": "Premium_LRS"
                },
                "diskSizeGB": 128
              }
            },
            "networkProfile" : {
              "networkInterfaces" : [
                {
                  "id" : "[resourceId('Microsoft.Network/networkInterfaces',
concat(variables('vmNames')[copyIndex()], '-nic'))]",
                  "properties": {
                    "primary": true
                  }
                }
              ]
            }
          }
        ]
      }
    }
  ]
}
```

7.10.17. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

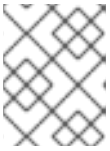
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.10.18. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

7.10.19. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名

要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

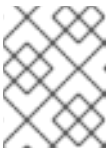
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.29.4
   master-1  Ready     master  73m  v1.29.4
   master-2  Ready     master  74m  v1.29.4
   worker-0  Ready     worker  11m  v1.29.4
   worker-1  Ready     worker  11m  v1.29.4
   ```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

7.10.20. Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用して、OpenShift Container Platform クラスターを Microsoft Azure にデプロイしています。
- OpenShift CLI (**oc**) をインストールすること。
- Azure CLI のインストールまたは更新を実行します。

手順

1. Ingress ルーターがロードバランサーを作成し、**EXTERNAL-IP** フィールドにデータを設定していることを確認します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.20.10  35.130.120.110
80:32288/TCP,443:31215/TCP  20
```

2. Ingress ルーター IP を変数としてエクスポートします。

```
$ export PUBLIC_IP_ROUTER=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

3. パブリック DNS ゾーンに ***.apps** レコードを追加します。
   a. このクラスターを新しいパブリックゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER} --ttl 300
```

   b. このクラスターを既存のパブリックゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n *.apps.${CLUSTER_NAME} -a ${PUBLIC_IP_ROUTER} --ttl 300
```

4. ***.apps** レコードをプライベート DNS ゾーンに追加します。
   a. 以下のコマンドを使用して ***.apps** レコードを作成します。

```
$ az network private-dns record-set a create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps --ttl 300
```

   b. 以下のコマンドを使用して ***.apps** レコードをプライベート DNS ゾーンに追加します。

```
$ az network private-dns record-set a add-record -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER}
```

ワイルドカードを使用する代わりに明示的なドメインを追加する場合は、クラスターのそれぞれの現行ルートのエントリーを作成できます。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}
{end}' routes
```

出力例

```
oauth-openshift.apps.cluster.basedomain.com
console-openshift-console.apps.cluster.basedomain.com
downloads-openshift-console.apps.cluster.basedomain.com
alertmanager-main-openshift-monitoring.apps.cluster.basedomain.com
prometheus-k8s-openshift-monitoring.apps.cluster.basedomain.com
```

7.10.21. ユーザーによってプロビジョニングされるインフラストラクチャーでの Azure インストールの実行

Microsoft Azure のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる Azure インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

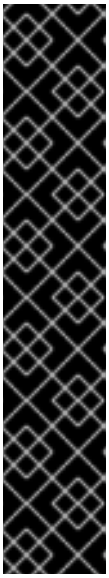
- クラスターのインストールを完了します。

  ```
  $ ./openshift-install --dir <installation_directory> wait-for install-complete 1
  ```

  出力例

  ```
  INFO Waiting up to 30m0s for the cluster to initialize...
  ```

  1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

7.10.22. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

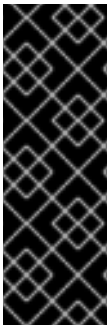
関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.11. ARM テンプレートを使用したクラスターの AZURE へのインストール

OpenShift Container Platform バージョン 4.16 では、独自に提供するインフラストラクチャーを使用して、Microsoft Azure にクラスターをインストールできます。

これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Azure Resource Manager (ARM) テンプレートが提供されます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の ARM テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

7.11.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。

- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 している。
- Azure CLI をダウンロードし、これをコンピューターにインストールしている。Azure ドキュメントの Install the Azure CLI を参照してください。以下のドキュメントについては、直近で Azure CLI のバージョン **2.38.0** を使用してテストされていますAzure CLI コマンドは、使用するバージョンによって動作が異なる場合があります。
- ご使用の環境でクラウド ID およびアクセス管理 (IAM) API にアクセスできない場合、または管理者レベルの認証情報シークレットを **kube-system** namespace に保存したくない場合は、管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法 を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

7.11.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.11.3. Azure プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするために Azure プロジェクトを設定する必要があります。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

7.11.3.1. Azure アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure コンポーネントを使用し、デフォルトの Azure サブスクリプションおよびサービス制限、クォータ、および制約 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。

重要

デフォルトの制限は、Free Trial や Pay-As-You-Go、および DV2、F、および G などのシリーズといったカテゴリータイプによって異なります。たとえば、Enterprise Agreement サブスクリプションのデフォルトは 350 コアです。

サブスクリプションタイプの制限を確認し、必要に応じて、デフォルトのクラスターを Azure にインストールする前にアカウントのクォータ制限を引き上げます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
vCPU	40	リージョンごとに 20	デフォルトのクラスターには 40 の vCPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップマシンは 4 vCPUS を使用する **Standard_D4s_v3** マシンを使用し、コントロールプレーンマシンは 8 vCPU を使用する **Standard_D8s_v3** 仮想マシンを使用し、さらにワーカーマシンは、4 vCPU を使用する **Standard_D4s_v3** 仮想マシンを使用するため、デフォルトクラスターには 40 の vCPU が必要になります。4 vCPU を使用するブートストラップノードの仮想マシンは、インストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。
OS ディスク	7		各クラスターマシンには、少なくとも 100 GB のストレージと 300 IOPS が必要です。これらはサポートされる最小の値ですが、実稼働クラスターおよび高負荷ワークロードがあるクラスターには、さらに高速なストレージが推奨されます。パフォーマンスを向上させるためのストレージ最適化について、詳しくは「スケーラビリティーとパフォーマンス」セクションの「ストレージの最適化」を参照してください。
VNet	1	リージョンごとに 1000	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
ネットワークインターフェイス	7	リージョンごとに 65,536	各デフォルトクラスターには、7 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。
ネットワークセキュリティーグループ	2	5000	各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。
ネットワークロードバランサー	3	リージョンごとに 1000	各クラスターは以下の ロードバランサー を作成します。 **default**: ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス **internal**: コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス **external**: コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス アプリケーションが追加の Kubernetes **LoadBalancer** サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
パブリック IP アドレス	3		2 つのパブリックロードバランサーのそれぞれはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。
プライベート IP アドレス	7		内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。
スポット VM vCPU (オプション)	0 スポット VM を設定する場合には、クラスターのコンピュートノードごとにスポット VM vCPU が 2 つ必要です。	リージョンごとに 20	これはオプションのコンポーネントです。スポット VM を使用するには、Azure のデフォルトの制限を最低でも、クラスター内のコンピュートノード数の 2 倍に増やす必要があります。 **注記** コントロールプレーンノードにスポット VM を使用することは推奨しません。

関連情報

- ストレージの最適化

7.11.3.2. Azure でのパブリック DNS ゾーンの設定

OpenShift Container Platform をインストールするには、使用する Microsoft Azure アカウントに、専用のパブリックホスト DNS ゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。このサービスは、クラスターへの外部接続のためのクラスター DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、Azure または別のソースから新規のものを取得できます。

注記

Azure 経由でドメインを購入する方法についての詳細は、Azure ドキュメントの Buy a custom domain name for Azure App Service を参照してください。

2. 既存のドメインおよびレジストラーを使用している場合、その DNS を Azure に移行します。Azure ドキュメントの Migrate an active DNS name to Azure App Service を参照してください。

3. ドメインの DNS を設定します。Azure ドキュメントの Tutorial: Host your domain in Azure DNS の手順に従い、ドメインまたはサブドメインのパブリックホストゾーンを作成し、 新規の権威ネームサーバーを抽出し、ドメインが使用するネームサーバーのレジストラーレコードを更新します。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。

この DNS ゾーンの作成例 を参照し、Azure の DNS ソリューションを確認することができます。

7.11.3.3. Azure アカウント制限の拡張

アカウントの制限を引き上げるには、Azure ポータルでサポートをリクエストします。

注記

サポートリクエストごとに 1 つの種類のクォータのみを増やすことができます。

手順

1. Azure ポータルの左端で **Help + support** をクリックします。

2. **New support request** をクリックしてから必要な値を選択します。

   a. **Issue type** リストから、**Service and subscription limits (quotas)**を選択します。

   b. **Subscription** リストから、変更するサブスクリプションを選択します。

   c. **Quota type** リストから、引き上げるクォータを選択します。たとえば、**Compute-VM (cores-vCPUs) subscription limit increases** を選択し、クラスターのインストールに必要な vCPU の数を増やします。

   d. **Next: Solutions** をクリックします。

3. **Problem Details**ページで、クォータの引き上げについての必要な情報を指定します。

   a. **Provide details**をクリックし、**Quota details**ウィンドウに必要な詳細情報を指定します。

   b. SUPPORT METHOD and CONTACT INFO セクションに、問題の重大度および問い合わせ先の詳細を指定します。

4. **Next: Review + create** をクリックしてから **Create** をクリックします。

7.11.3.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認しま

す。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

7.11.3.5. サブスクリプション ID とテナント ID の記録

インストールプログラムには、Azure アカウントに関連付けられたサブスクリプション ID とテナント ID が必要です。Azure CLI を使用してこの情報を収集できます。

前提条件

- Azure CLI をインストールまたは更新している。

手順

1. 次のコマンドを実行して、Azure CLI にログインします。

```
$ az login
```

2. 適切なサブスクリプションを使用していることを確認してください。
   a. 次のコマンドを実行して、利用可能なサブスクリプションのリストを表示します。

```
$ az account list --refresh
```

出力例

```
[
  {
    "cloudName": "AzureCloud",
    "id": "8xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "isDefault": true,
    "name": "Subscription Name 1",
    "state": "Enabled",
    "tenantId": "6xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  },
  {
    "cloudName": "AzureCloud",
    "id": "9xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "isDefault": false,
    "name": "Subscription Name 2",
    "state": "Enabled",
    "tenantId": "7xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "user": {
      "name": "you2@example.com",
      "type": "user"
    }
  }
]
```

b. 次のコマンドを実行して、アクティブなアカウントの詳細を表示し、これが使用するサブスクリプションであることを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "8xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "isDefault": true,
  "name": "Subscription Name 1",
  "state": "Enabled",
  "tenantId": "6xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 適切なサブスクリプションを使用していない場合は、以下を行います。

a. 次のコマンドを実行して、アクティブなサブスクリプションを変更します。

```
$ az account set -s <subscription_id>
```

b. 次のコマンドを実行して、必要なサブスクリプションを使用していることを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "9xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "isDefault": true,
  "name": "Subscription Name 2",
  "state": "Enabled",
  "tenantId": "7xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "user": {
    "name": "you2@example.com",
    "type": "user"
  }
}
```

4. 出力から **id** および **tenantId** パラメーターの値を記録します。OpenShift Container Platform クラスターをインストールするには、これらの値が必要です。

7.11.3.6. Azure リソースにアクセスするためにサポートされている ID

OpenShift Container Platform クラスターでは、Azure リソースを作成および管理するために Azure ID が必要です。したがって、インストールを完了するには、次のいずれかのタイプの ID が必要です。

- サービスプリンシパル

- システムが割り当てたマネージド ID
- ユーザーが割り当てたマネージド ID

7.11.3.7. ユーザーがプロビジョニングするインフラストラクチャーに必要な Azure アクセス許可

インストールプログラムでは、クラスターをデプロイし、日常の操作を維持するために必要なパーミッションを持つ Azure サービスプリンシパルまたはマネージド ID にアクセスする必要があります。これらパーミッションは、ID に関連付けられた Azure サブスクリプションに付与する必要があります。

以下のオプションを使用できます。

- ID に **Contributor** ロールと **User Access Administrator** ロールを割り当てることができます。これらのロールを割り当てるのが、必要な権限をすべて付与する最も簡単な方法です。
ロールの割り当ての詳細は、Azure portal を使用した Azure リソースへのアクセスの管理 に関する Azure ドキュメントを参照してください。
- 組織のセキュリティーポリシーで、より制限的なアクセス許可のセットが必要な場合は、必要なアクセス許可を持つ カスタムロール を作成できます。

Microsoft Azure で OpenShift Container Platform クラスターを作成するには、以下のアクセス許可が必要です。

例7.64 承認リソースを作成するために必要な権限

- **Microsoft.Authorization/policies/audit/action**
- **Microsoft.Authorization/policies/auditIfNotExists/action**
- **Microsoft.Authorization/roleAssignments/read**
- **Microsoft.Authorization/roleAssignments/write**

例7.65 コンピューティングリソースの作成に必要な権限

- **Microsoft.Compute/images/read**
- **Microsoft.Compute/images/write**
- **Microsoft.Compute/images/delete**
- **Microsoft.Compute/availabilitySets/read**
- **Microsoft.Compute/disks/beginGetAccess/action**
- **Microsoft.Compute/disks/delete**
- **Microsoft.Compute/disks/read**
- **Microsoft.Compute/disks/write**
- **Microsoft.Compute/galleries/images/read**
- **Microsoft.Compute/galleries/images/versions/read**

- **Microsoft.Compute/galleries/images/versions/write**
- **Microsoft.Compute/galleries/images/write**
- **Microsoft.Compute/galleries/read**
- **Microsoft.Compute/galleries/write**
- **Microsoft.Compute/snapshots/read**
- **Microsoft.Compute/snapshots/write**
- **Microsoft.Compute/snapshots/delete**
- **Microsoft.Compute/virtualMachines/delete**
- **Microsoft.Compute/virtualMachines/powerOff/action**
- **Microsoft.Compute/virtualMachines/read**
- **Microsoft.Compute/virtualMachines/write**
- **Microsoft.Compute/virtualMachines/deallocate/action**

例7.66 ID 管理リソースを作成するために必要なアクセス許可

- **Microsoft.ManagedIdentity/userAssignedIdentities/assign/action**
- **Microsoft.ManagedIdentity/userAssignedIdentities/read**
- **Microsoft.ManagedIdentity/userAssignedIdentities/write**

例7.67 ネットワークリソースの作成に必要な権限

- **Microsoft.Network/dnsZones/A/write**
- **Microsoft.Network/dnsZones/CNAME/write**
- **Microsoft.Network/dnszones/CNAME/read**
- **Microsoft.Network/dnszones/read**
- **Microsoft.Network/loadBalancers/backendAddressPools/join/action**
- **Microsoft.Network/loadBalancers/backendAddressPools/read**
- **Microsoft.Network/loadBalancers/backendAddressPools/write**
- **Microsoft.Network/loadBalancers/read**
- **Microsoft.Network/loadBalancers/write**
- **Microsoft.Network/networkInterfaces/delete**
- **Microsoft.Network/networkInterfaces/join/action**

- **Microsoft.Network/networkInterfaces/read**
- **Microsoft.Network/networkInterfaces/write**
- **Microsoft.Network/networkSecurityGroups/join/action**
- **Microsoft.Network/networkSecurityGroups/read**
- **Microsoft.Network/networkSecurityGroups/securityRules/delete**
- **Microsoft.Network/networkSecurityGroups/securityRules/read**
- **Microsoft.Network/networkSecurityGroups/securityRules/write**
- **Microsoft.Network/networkSecurityGroups/write**
- **Microsoft.Network/privateDnsZones/A/read**
- **Microsoft.Network/privateDnsZones/A/write**
- **Microsoft.Network/privateDnsZones/A/delete**
- **Microsoft.Network/privateDnsZones/SOA/read**
- **Microsoft.Network/privateDnsZones/read**
- **Microsoft.Network/privateDnsZones/virtualNetworkLinks/read**
- **Microsoft.Network/privateDnsZones/virtualNetworkLinks/write**
- **Microsoft.Network/privateDnsZones/write**
- **Microsoft.Network/publicIPAddresses/delete**
- **Microsoft.Network/publicIPAddresses/join/action**
- **Microsoft.Network/publicIPAddresses/read**
- **Microsoft.Network/publicIPAddresses/write**
- **Microsoft.Network/virtualNetworks/join/action**
- **Microsoft.Network/virtualNetworks/read**
- **Microsoft.Network/virtualNetworks/subnets/join/action**
- **Microsoft.Network/virtualNetworks/subnets/read**
- **Microsoft.Network/virtualNetworks/subnets/write**
- **Microsoft.Network/virtualNetworks/write**

例7.68 リソースの正常性をチェックするために必要なアクセス許可

- **Microsoft.Resourcehealth/healthevent/Activated/action**
- **Microsoft.Resourcehealth/healthevent/InProgress/action**

- **Microsoft.Resourcehealth/healthevent/Pending/action**
- **Microsoft.Resourcehealth/healthevent/Resolved/action**
- **Microsoft.Resourcehealth/healthevent/Updated/action**

例7.69 リソースグループの作成に必要なアクセス許可

- **Microsoft.Resources/subscriptions/resourceGroups/read**
- **Microsoft.Resources/subscriptions/resourcegroups/write**

例7.70 リソースタグの作成に必要なアクセス許可

- **Microsoft.Resources/tags/write**

例7.71 ストレージリソースの作成に必要な権限

- **Microsoft.Storage/storageAccounts/blobServices/read**
- **Microsoft.Storage/storageAccounts/blobServices/containers/write**
- **Microsoft.Storage/storageAccounts/fileServices/read**
- **Microsoft.Storage/storageAccounts/fileServices/shares/read**
- **Microsoft.Storage/storageAccounts/fileServices/shares/write**
- **Microsoft.Storage/storageAccounts/fileServices/shares/delete**
- **Microsoft.Storage/storageAccounts/listKeys/action**
- **Microsoft.Storage/storageAccounts/read**
- **Microsoft.Storage/storageAccounts/write**

例7.72 デプロイメントの作成に必要な権限

- **Microsoft.Resources/deployments/read**
- **Microsoft.Resources/deployments/write**
- **Microsoft.Resources/deployments/validate/action**
- **Microsoft.Resources/deployments/operationstatuses/read**

例7.73 コンピュートリソースを作成するためのオプションのアクセス許可

- **Microsoft.Compute/availabilitySets/write**

例7.74 Marketplace 仮想マシンリソースを作成するためのオプションのアクセス許可

- **Microsoft.MarketplaceOrdering/offertypes/publishers/offers/plans/agreements/read**
- **Microsoft.MarketplaceOrdering/offertypes/publishers/offers/plans/agreements/write**

例7.75 ユーザー管理の暗号化を有効にするためのオプションのアクセス許可

- **Microsoft.Compute/diskEncryptionSets/read**
- **Microsoft.Compute/diskEncryptionSets/write**
- **Microsoft.Compute/diskEncryptionSets/delete**
- **Microsoft.KeyVault/vaults/read**
- **Microsoft.KeyVault/vaults/write**
- **Microsoft.KeyVault/vaults/delete**
- **Microsoft.KeyVault/vaults/deploy/action**
- **Microsoft.KeyVault/vaults/keys/read**
- **Microsoft.KeyVault/vaults/keys/write**
- **Microsoft.Features/providers/features/register/action**

Microsoft Azure で OpenShift Container Platform クラスターを削除するには、以下のアクセス許可が必要です。

例7.76 承認リソースを削除するために必要な権限

- **Microsoft.Authorization/roleAssignments/delete**

例7.77 コンピューティングリソースを削除するために必要な権限

- **Microsoft.Compute/disks/delete**
- **Microsoft.Compute/galleries/delete**
- **Microsoft.Compute/galleries/images/delete**
- **Microsoft.Compute/galleries/images/versions/delete**
- **Microsoft.Compute/virtualMachines/delete**
- **Microsoft.Compute/images/delete**

例7.78 Required permissions for deleting identity management resources

- **Microsoft.ManagedIdentity/userAssignedIdentities/delete**

例7.79 ネットワークリソースを削除するために必要な権限

- Microsoft.Network/dnszones/read
- Microsoft.Network/dnsZones/A/read
- Microsoft.Network/dnsZones/A/delete
- Microsoft.Network/dnsZones/CNAME/read
- Microsoft.Network/dnsZones/CNAME/delete
- Microsoft.Network/loadBalancers/delete
- Microsoft.Network/networkInterfaces/delete
- Microsoft.Network/networkSecurityGroups/delete
- Microsoft.Network/privateDnsZones/read
- Microsoft.Network/privateDnsZones/A/read
- Microsoft.Network/privateDnsZones/delete
- Microsoft.Network/privateDnsZones/virtualNetworkLinks/delete
- Microsoft.Network/publicIPAddresses/delete
- Microsoft.Network/virtualNetworks/delete

例7.80 リソースの正常性をチェックするために必要なアクセス許可

- Microsoft.Resourcehealth/healthevent/Activated/action
- Microsoft.Resourcehealth/healthevent/Resolved/action
- Microsoft.Resourcehealth/healthevent/Updated/action

例7.81 リソースグループを削除するために必要なアクセス許可

- Microsoft.Resources/subscriptions/resourcegroups/delete

例7.82 ストレージリソースを削除するために必要な権限

- Microsoft.Storage/storageAccounts/delete
- Microsoft.Storage/storageAccounts/listKeys/action

注記

Azure に OpenShift Container Platform をインストールするには、リソースグループの作成に関連するアクセス許可をサブスクリプションに限定する必要があります。リソースグループが作成されたら、作成されたリソースグループに残りのアクセス許可のスコープを設定できます。パブリック DNS ゾーンが別のリソースグループに存在する場合は、ネットワーク DNS ゾーンに関連するアクセス許可を常にサブスクリプションに適用する必要があります。

OpenShift Container Platform クラスターを削除するときに、すべてのパーミッションをサブスクリプションに限定できます。

7.11.3.8. Azure マネージド ID の使用

インストールプログラムでは、インストールを完了するために Azure ID が必要です。システム割り当てまたはユーザー割り当てのマネージド ID を使用できます。

マネージド ID を使用できない場合は、サービスプリンシパルを使用できます。

手順

1. システム割り当てのマネージド ID を使用している場合は、インストールプログラムを実行する仮想マシン上でそれを有効にします。
2. ユーザーが割り当てたマネージド ID を使用している場合は以下を行います。
   a. これを、インストールプログラムを実行する仮想マシンに割り当てます。
   b. クライアント ID を記録します。この値は、クラスターをインストールするときに必要になります。
   ユーザー割り当てマネージド ID の詳細を表示する場合については、Microsoft Azure ドキュメントで ユーザー割り当てマネージド ID のリスト を参照してください。
3. 必要なパーミッションがマネージド ID に割り当てられていることを確認します。

7.11.3.9. サービスプリンシパルの作成

インストールプログラムでは、インストールを完了するために Azure ID が必要です。サービスプリンシパルを使用できます。

サービスプリンシパルを使用できない場合は、マネージド ID を使用できます。

前提条件

- Azure CLI をインストールまたは更新している。
- Azure サブスクリプション ID がある。
- サービスプリンシパルに **Contributor** ロールおよび **User Administrator Access** ロールを割り当てない場合は、必要な Azure アクセス許可を持つカスタムロールを作成しています。

手順

1. 次のコマンドを実行して、アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role <role_name> \ 1
```

```
  --name <service_principal> \ 2
  --scopes /subscriptions/<subscription_id> 3
```

1 ロール名を定義します。**Contributor** ロールを使用するか、必要なアクセス許可を含むカスタムロールを指定できます。

2 サービスプリンシパル名を定義します。

3 サブスクリプション ID を指定します。

出力例

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "axxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "displayName": <service_principal>",
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

2. 出力から **appId** パラメーターと **password** パラメーターの値を記録します。クラスターをインストールするときにこれらの値が必要です。

3. **Contributor** ロールをサービスプリンシパルに適用した場合は、次のコマンドを実行して **User Administrator Access** ロールを割り当てます。

```
$ az role assignment create --role "User Access Administrator" \
  --assignee-object-id $(az ad sp show --id <appId> --query id -o tsv) 1
  --scope /subscriptions/<subscription_id> 2
```

1 サービスプリンシパルの **appId** パラメーター値を指定します。

2 サブスクリプション ID を指定します。

関連情報

- CCO モードの詳細は、Cloud Crednetial Operator について を参照してください。

7.11.3.10. サポート対象の Azure リージョン

インストールプログラムは、サブスクリプションに基づいて利用可能な Microsoft Azure リージョンのリストを動的に生成します。

サポート対象の Azure パブリックリージョン

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)

- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)
- **canadaeast** (Canada East)
- **centralindia** (Central India)
- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)
- **eastus2** (East US 2)
- **francecentral** (France Central)
- **germanywestcentral** (Germany West Central)
- **israelcentral** (イスラエル中央)
- **italynorth** (イタリア北部)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **mexicocentral** (Mexico Central)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)
- **norwayeast** (Norway East)
- **polandcentral** (ポーランド中央)
- **qarcentral** (カタール中部)
- **southafricanorth** (South Africa North)
- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)
- **swedencentral** (スウェーデン中央)
- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)

- **uksouth** (UK South)
- **ukwest** (UK West)
- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)
- **westus3**(West US 3)

サポート対象の Azure Government リージョン
以下の Microsoft Azure Government (MAG) リージョンのサポートが OpenShift Container Platform バージョン 4.6 に追加されています。

- **usgovtexas** (US Gov Texas)
- **usgovvirginia** (US Gov Virginia)

Azure ドキュメント の利用可能なすべての MAG リージョンを参照できます。他の提供される MAG リージョンは OpenShift Container Platform で機能することが予想されますが、まだテストされていません。

7.11.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

7.11.4.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表7.25 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。

ホスト	説明
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

7.11.4.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表7.26 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

重要

premiumIO パラメーターが **true** に設定されている Azure 仮想マシンを使用する必要があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

7.11.4.3. Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure インスタンスタイプは OpenShift Container Platform でテストされています。

例7.83 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***

- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

7.11.4.4. 64 ビット ARM インフラストラクチャー上の Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure Azure64 インスタンスタイプは OpenShift Container Platform でテストされています。

例7.84 64 ビット ARM アーキテクチャーに基づくマシンタイプ

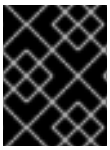
- **c6g.***
- **m6g.***

7.11.5. Azure Marketplace オファリングの使用

Azure Marketplace を使用すると、OpenShift Container Platform クラスターをデプロイできます。これは、Azure を通じて従量課金制 (時間単位、コア単位) で請求され、Red Hat の直接サポートも受けることができます。

Azure Marketplace オファリングを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に Azure Marketplace イメージを取得する必要があります。インストールプログラムは、このイメージを使用してワーカーまたはコントロールプレーンノードをデプロイします。イメージを取得するときは、次の点を考慮してください。

- イメージは同じですが、Azure Marketplace のパブリシャーは地域によって異なります。北米にお住まいの場合は、**redhat** をパブリッシャーとして指定してください。EMEA にお住まいの場合は、**redhat-limited** をパブリッシャーとして指定してください。
- このオファーには、**rh-ocp-worker** SKU と **rh-ocp-worker-gen1** SKU が含まれています。**rh-ocp-worker** SKU は、Hyper-V 世代のバージョン 2 VM イメージを表します。OpenShift Container Platform で使用されるデフォルトのインスタンスタイプは、バージョン 2 と互換性があります。バージョン 1 のみと互換性のあるインスタンスタイプを使用する場合は、**rh-ocp-worker-gen1** SKU に関連付けられたイメージを使用します。**rh-ocp-worker-gen1** SKU は、Hyper-V バージョン 1 VM イメージを表します。



重要

Azure マーケットプレイスを使用したイメージのインストールは、64 ビット ARM インスタンスを備えたクラスターではサポートされていません。

前提条件

- Azure CLI クライアント **(az)** をインストールしている。
- お客様の Azure アカウントにはオファーのエンタイトルメントがあり、Azure CLI クライアントを使用してこのアカウントにログインしている。

手順

1. 以下のいずれかのコマンドを実行して、利用可能なすべての OpenShift Container Platform イメージを表示します。
   - 北米:

     ```
     $  az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
     ```

     出力例

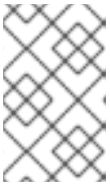
     ```
     Offer          Publisher       Sku                Urn                                                                  Version
     -------------  --------------  -----------------  -------------------------------------------------------------------
     ----  -----------------
     rh-ocp-worker  RedHat          rh-ocp-worker      RedHat:rh-ocp-worker:rh-ocp-
     worker:413.92.2023101700           413.92.2023101700
     rh-ocp-worker  RedHat          rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-
     gen1:413.92.2023101700       413.92.2023101700
     ```

   - EMEA:

     ```
     $  az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
     ```

     出力例

     ```
     Offer          Publisher       Sku                Urn
     Version
     -------------  --------------  -----------------  -------------------------------------------------------------------
     ----          -----------------
     rh-ocp-worker  redhat-limited  rh-ocp-worker      redhat-limited:rh-ocp-worker:rh-ocp-
     worker:413.92.2023101700           413.92.2023101700
     rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
     worker-gen1:413.92.2023101700       413.92.2023101700
     ```

     注記

     コンピュートおよびコントロールプレーンノードで利用可能な最新のイメージを使用します。必要に応じて、VM はインストールプロセスの一部として自動的にアップグレードされます。

2. 次のいずれかのコマンドを実行して、オファーのイメージを調べます。
   - 北米:

     ```
     $ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

- EMEA:

```
$ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

3. 次のコマンドのいずれかを実行して、オファーの条件を確認します。

- 北米:

```
$ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. 次のコマンドのいずれかを実行して、オファリングの条件に同意します。

- 北米:

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. オファーのイメージの詳細を記録します。Azure Resource Manager (ARM)テンプレートを使用してコンピュートノードをデプロイする場合：

a. **id** パラメーターを削除し、オファーの値を使用して、**offer**、**publisher**、**sku**、および **version** パラメーターを追加して、**storageProfile.imageReference** を更新します。

b. 仮想マシン (VM) の **plan** を指定します。

更新された storageProfile.imageReference オブジェクトと指定された plan を含む 06_workers.json ARM テンプレートの例

```
...
  "plan" : {
    "name": "rh-ocp-worker",
    "product": "rh-ocp-worker",
    "publisher": "redhat"
  },
  "dependsOn" : [
    "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')
[copyIndex()], '-nic'))]"
  ],
  "properties" : {
...
  "storageProfile": {
    "imageReference": {
    "offer": "rh-ocp-worker",
    "publisher": "redhat",
    "sku": "rh-ocp-worker",
    "version": "413.92.2023101700"
    }
```

```
      ...
    }
...
    }
```

7.11.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

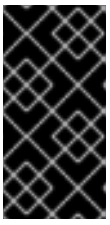
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.11.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

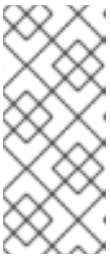
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

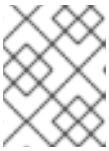
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

7.11.8. Azure のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Microsoft Azure にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

7.11.8.1. オプション: 別個の /var パーティションの作成

OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

7.11.8.2. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルを使用してクラスターをインストールしている場合は、そのアプリケーション ID とパスワードが必要です。
- システムが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、インストールプログラムを実行する仮想マシン上でそれが有効になっています。
- ユーザーが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、次の前提条件を満たしている必要があります。
  - そのクライアント ID がある。
  - これは、インストールプログラムを実行する仮想マシンに割り当てられている。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルまたはマネージド ID を使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。

2. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。
インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。

iii. サブスクリプションの次の Azure パラメーター値を入力します。

- **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
- **azure tenant id** テナント ID を入力します。

iv. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアント ID** の入力を求められたら、次のいずれかを行います。

- サービスプリンシパルを使用している場合は、そのアプリケーション ID を入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、そのクライアント ID を指定します。

v. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアントシークレット** の入力を求められたら、次のいずれかを実行します。

- サービスプリンシパルを使用している場合は、そのパスワードを入力します。

- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

vi. クラスターをデプロイするリージョンを選択します。

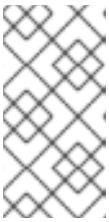
vii. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

viii. クラスターの記述名を入力します。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

3. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。



注記

3 ノードクラスターをインストールする場合は、必ず **compute.replicas** パラメーターを **0** に設定してください。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細については、「Azure に 3 ノードクラスターをインストールする」を参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの **~/.azure/** ディレクトリーに保存します。これにより、インストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作成するときにプロファイルをロードできるようになります。

7.11.8.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

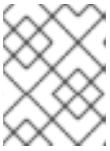
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

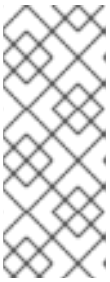
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-**

bundle 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

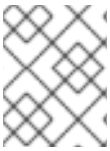
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

7.11.8.4. ARM テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Microsoft Azure で実行するのに役立つ指定の Azure Resource Manager (ARM) テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。



注記

特定の ARM テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. 提供される ARM テンプレートで使用される **install-config.yaml** にある一般的な変数をエクスポートします。

```
$ export CLUSTER_NAME=<cluster_name>1
$ export AZURE_REGION=<azure_region>2
$ export SSH_KEY=<ssh_key>3
```

```
$ export BASE_DOMAIN=<base_domain> 4
$ export BASE_DOMAIN_RESOURCE_GROUP=<base_domain_resource_group> 5
```

1. **install-config.yaml** ファイルからの **.metadata.name** 属性の値。
2. クラスターをデプロイするリージョン (例: **centralus**)。これは、**install-config.yaml** ファイルからの **.platform.azure.region** 属性の値です。
3. 文字列としての SSH RSA 公開鍵ファイル。SSH キーは、スペースが含まれているために引用符で囲む必要があります。これは、**install-config.yaml** ファイルからの **.sshKey** 属性の値です。
4. クラスターをデプロイするベースドメイン。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。これは、**install-config.yaml** からの **.baseDomain** 属性の値です。
5. パブリック DNS ゾーンが存在するリソースグループ。これは、**install-config.yaml** ファイルからの **.platform.azure.baseDomainResourceGroupName** 属性の値です。

以下に例を示します。

```
$ export CLUSTER_NAME=test-cluster
$ export AZURE_REGION=centralus
$ export SSH_KEY="ssh-rsa xxx/xxx/xxx= user@email.com"
$ export BASE_DOMAIN=example.com
$ export BASE_DOMAIN_RESOURCE_GROUP=ocp-cluster
```

2. kubeadmin 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

7.11.8.5. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

4. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

重要

ユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールするときに **MachineAPI** 機能を無効にした場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除する必要があります。そうしないと、クラスターのインストールに失敗します。

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. ユーザーによってプロビジョニングされるインフラストラクチャーで Azure を設定する場合、Azure Resource Manager (ARM) テンプレートで後に使用するためにマニフェストファイルに定義された一般的な変数の一部をエクスポートする必要があります。

   a. 以下のコマンドを使用してインフラストラクチャー ID をエクスポートします。

   ```
   $ export INFRA_ID=<infra_id> 1
   ```

   1 OpenShift Container Platform クラスターには、**<cluster_name>-<random_string>** の形式の識別子 (**INFRA_ID**) が割り当てられます。これは、提供される ARM テンプレートを使用して作成されるほとんどのリソースのベース名として使用されます。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.infrastructureName** 属性の値です。

   b. 以下のコマンドを使用してリソースグループをエクスポートします。

   ```
   $ export RESOURCE_GROUP=<resource_group> 1
   ```

   1 この Azure デプロイメントで作成されたすべてのリソースは、リソースグループ の一部として存在します。リソースグループ名は、**<cluster_name>-<random_string>-rg** 形式の **INFRA_ID** をベースとしています。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.platformStatus.azure.resourceGroupName** 属性の値です。

8. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

   ```
   $ ./openshift-install create ignition-configs --dir <installation_directory> 1
   ```

   1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

   Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

   ```
   .
   ├── auth
   │   ├── kubeadmin-password
   │   └── kubeconfig
   ├── bootstrap.ign
   ├── master.ign
   ├── metadata.json
   └── worker.ign
   ```

7.11.9. Azure リソースグループの作成

Microsoft Azure リソースグループ およびリソースグループのアイデンティティーを作成する必要があります。これらはいずれも Azure での OpenShift Container Platform クラスターのインストール時に使用されます。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. サポートされる Azure リージョンにリソースグループを作成します。

```
$ az group create --name ${RESOURCE_GROUP} --location ${AZURE_REGION}
```

2. リソースグループの Azure アイデンティティーを作成します。

```
$ az identity create -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity
```

これは、クラスター内の Operator に必要なアクセスを付与するために使用されます。たとえば、これにより Ingress Operator はパブリック IP およびそのロードバランサーを作成できます。Azure アイデンティティーをロールに割り当てる必要があります。

3. Contributor ロールを Azure アイデンティティーに付与します。

a. Azure ロールの割り当てで必要な以下の変数をエクスポートします。

```
$ export PRINCIPAL_ID=`az identity show -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity --query principalId --out tsv`
```

```
$ export RESOURCE_GROUP_ID=`az group show -g ${RESOURCE_GROUP} --query id --out tsv`
```

b. Contributor ロールをアイデンティティーに割り当てます。

```
$ az role assignment create --assignee "${PRINCIPAL_ID}" --role 'Contributor' --scope "${RESOURCE_GROUP_ID}"
```

注記

必要なすべてのアクセス許可を持つカスタムロールを ID に割り当てる場合は、次のコマンドを実行します。

```
$ az role assignment create --assignee "${PRINCIPAL_ID}" --role <custom_role> \ 1
--scope "${RESOURCE_GROUP_ID}"
```

1 カスタムロール名を指定します。

7.11.10. RHCOS クラスターイメージおよびブートストラップ Ignition 設定ファイルのアップロード

Azure クライアントは、ローカルに存在するファイルに基づくデプロイメントをサポートしていません。RHCOS 仮想ハードディスク (VHD) クラスターイメージとブートストラップ Ignition 設定ファイルをコピーしてストレージコンテナーに保存し、デプロイメント中にアクセスできるようにする必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. VHD クラスターイメージを保存するために Azure ストレージアカウントを作成します。

```
$ az storage account create -g ${RESOURCE_GROUP} --location ${AZURE_REGION} --
name ${CLUSTER_NAME}sa --kind Storage --sku Standard_LRS
```

警告

Azure ストレージアカウント名は 3 文字から 24 文字の長さで、数字および小文字のみを使用する必要があります。**CLUSTER_NAME** 変数がこれらの制限に準拠しない場合、Azure ストレージアカウント名を手動で定義する必要があります。Azure ストレージアカウント名の制限についての詳細は、Azure ドキュメントの Resolve errors for storage account names を参照してください。

2. ストレージアカウントキーを環境変数としてエクスポートします。

```
$ export ACCOUNT_KEY=`az storage account keys list -g ${RESOURCE_GROUP} --
account-name ${CLUSTER_NAME}sa --query "[0].value" -o tsv`
```

3. RHCOS VHD の URL を環境変数にエクスポートします。

```
$ export VHD_URL=`openshift-install coreos print-stream-json | jq -r '.architectures.
<architecture>."rhel-coreos-extensions"."azure-disk".url'`
```

ここでは、以下のようになります。

<architecture>
アーキテクチャーを指定します。有効な値は **x86_64** または **aarch64** です。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージを指定する必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

4. VHD のストレージコンテナーを作成します。

```
$ az storage container create --name vhd --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY}
```

5. ローカル VHD を blob にコピーします。

```
$ az storage blob copy start --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} --destination-blob "rhcos.vhd" --destination-container vhd --source-uri "${VHD_URL}"
```

6. blob ストレージコンテナーを作成し、生成された **bootstrap.ign** ファイルをアップロードします。

```
$ az storage container create --name files --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY}
```

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -c "files" -f "<installation_directory>/bootstrap.ign" -n "bootstrap.ign"
```

7.11.11. DNS ゾーンの作成例

DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターに必要です。シナリオに適した DNS ストラテジーを選択する必要があります。

この例の場合、Azure の DNS ソリューション が使用されるため、外部 (インターネット) の可視性のために新規パブリック DNS ゾーンと、内部クラスターの解決用にプライベート DNS ゾーンが作成されます。

注記

パブリック DNS ゾーンは、クラスターデプロイメントと同じリソースグループに存在している必要はなく、必要なベースドメイン用にすでに組織内に存在している可能性があります。その場合、パブリック DNS ゾーンの作成を省略できます。先に生成したインストール設定がこのシナリオに基づいていることを確認してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. **BASE_DOMAIN_RESOURCE_GROUP** 環境変数でエクスポートされたリソースグループに、新規のパブリック DNS ゾーンを作成します。

```
$ az network dns zone create -g ${BASE_DOMAIN_RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

   すでに存在するパブリック DNS ゾーンを使用している場合は、この手順を省略できます。

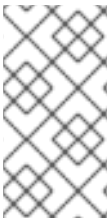
2. このデプロイメントの残りの部分と同じリソースグループにプライベート DNS ゾーンを作成します。

```
$ az network private-dns zone create -g ${RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

Azure でのパブリック DNS ゾーンの設定 についてのセクションを参照してください。

7.11.12. Azure での VNet の作成

OpenShift Container Platform クラスター用に Microsoft Azure で使用する仮想ネットワーク (VNet) を作成する必要があります。各種の要件を満たすように VPC をカスタマイズできます。VNet を作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

提供される ARM テンプレートを使用して Azure インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VNet の ARM テンプレート**セクションからテンプレートをコピーし、これを **01_vnet.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要な VNet について記述しています。

2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/01_vnet.json" \
  --parameters baseName="${INFRA_ID}" 1
```

   1 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. VNet テンプレートをプライベート DNS ゾーンにリンクします。

```
$ az network private-dns link vnet create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n ${INFRA_ID}-network-link -v "${INFRA_ID}-vnet"
-e false
```

7.11.12.1. VNet の ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な VNet をデプロイすることができます。

例7.85 01_vnet.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "addressPrefix" : "10.0.0.0/16",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetPrefix" : "10.0.0.0/24",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetPrefix" : "10.0.1.0/24",
    "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/virtualNetworks",
      "name" : "[variables('virtualNetworkName')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/networkSecurityGroups/', variables('clusterNsgName'))]"
      ],
      "properties" : {
        "addressSpace" : {
          "addressPrefixes" : [
            "[variables('addressPrefix')]"
          ]
        },
        "subnets" : [
          {
            "name" : "[variables('masterSubnetName')]",
            "properties" : {
              "addressPrefix" : "[variables('masterSubnetPrefix')]",
              "serviceEndpoints": [],
```

```
            "networkSecurityGroup" : {
              "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
            }
          }
        },
        {
          "name" : "[variables('nodeSubnetName')]",
          "properties" : {
            "addressPrefix" : "[variables('nodeSubnetPrefix')]",
            "serviceEndpoints": [],
            "networkSecurityGroup" : {
              "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
            }
          }
        }
      ]
    }
  },
  {
    "type" : "Microsoft.Network/networkSecurityGroups",
    "name" : "[variables('clusterNsgName')]",
    "apiVersion" : "2018-10-01",
    "location" : "[variables('location')]",
    "properties" : {
      "securityRules" : [
        {
          "name" : "apiserver_in",
          "properties" : {
            "protocol" : "Tcp",
            "sourcePortRange" : "*",
            "destinationPortRange" : "6443",
            "sourceAddressPrefix" : "*",
            "destinationAddressPrefix" : "*",
            "access" : "Allow",
            "priority" : 101,
            "direction" : "Inbound"
          }
        }
      ]
    }
  }
 ]
}
```

7.11.13. Azure インフラストラクチャー用の RHCOS クラスターイメージのデプロイ

OpenShift Container Platform ノードに Microsoft Azure 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

前提条件

- Azure アカウントを設定します。

- クラスターの Ignition 設定ファイルを生成します。
- RHCOS 仮想ハードディスク (VHD) クラスターイメージを Azure ストレージコンテナーに保存します。
- ブートストラップ Ignition 設定ファイルを Azure ストレージコンテナーに保存します。

手順

1. 本トピックの **イメージストレージの ARM テンプレート**セクションからテンプレートをコピーし、これを **02_storage.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なイメージストレージ について記述しています。
2. RHCOS VHD blob URL を変数としてエクスポートします。

```
$ export VHD_BLOB_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -o tsv`
```

3. クラスターイメージのデプロイ

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/02_storage.json" \
  --parameters vhdBlobURL="${VHD_BLOB_URL}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameters storageAccount="${CLUSTER_NAME}sa" \ 3
  --parameters architecture="<architecture>" 4
```

1. マスターマシンおよびワーカーマシンを作成するために使用される RHCOS VHD の blob URL。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. Azure ストレージアカウントの名前。
4. システムアーキテクチャーを指定します。有効な値は、**x64** (デフォルト) または **Arm64** です。

7.11.13.1. イメージストレージの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な保存された Red Hat Enterprise Linux CoreOS (RHCOS) をデプロイすることができます。

例7.86 02_storage.json ARM テンプレート

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "architecture": {
      "type": "string",
      "metadata": {
```

```
        "description": "The architecture of the Virtual Machines"
      },
      "defaultValue": "x64",
      "allowedValues": [
        "Arm64",
        "x64"
      ]
    },
    "baseName": {
      "type": "string",
      "minLength": 1,
      "metadata": {
        "description": "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "storageAccount": {
      "type": "string",
      "metadata": {
        "description": "The Storage Account name"
      }
    },
    "vhdBlobURL": {
      "type": "string",
      "metadata": {
        "description": "URL pointing to the blob where the VHD to be used to create master and
worker machines is located"
      }
    }
  },
  "variables": {
    "location": "[resourceGroup().location]",
    "galleryName": "[concat('gallery_', replace(parameters('baseName'), '-', '_'))]",
    "imageName": "[parameters('baseName')]",
    "imageNameGen2": "[concat(parameters('baseName'), '-gen2')]",
    "imageRelease": "1.0.0"
  },
  "resources": [
    {
      "apiVersion": "2021-10-01",
      "type": "Microsoft.Compute/galleries",
      "name": "[variables('galleryName')]",
      "location": "[variables('location')]",
      "resources": [
        {
          "apiVersion": "2021-10-01",
          "type": "images",
          "name": "[variables('imageName')]",
          "location": "[variables('location')]",
          "dependsOn": [
            "[variables('galleryName')]"
          ],
          "properties": {
            "architecture": "[parameters('architecture')]",
            "hyperVGeneration": "V1",
            "identifier": {
              "offer": "rhcos",
```

```
            "publisher": "RedHat",
            "sku": "basic"
          },
          "osState": "Generalized",
          "osType": "Linux"
        },
        "resources": [
          {
            "apiVersion": "2021-10-01",
            "type": "versions",
            "name": "[variables('imageRelease')]",
            "location": "[variables('location')]",
            "dependsOn": [
              "[variables('imageName')]"
            ],
            "properties": {
              "publishingProfile": {
                "storageAccountType": "Standard_LRS",
                "targetRegions": [
                  {
                    "name": "[variables('location')]",
                    "regionalReplicaCount": "1"
                  }
                ]
              },
              "storageProfile": {
                "osDiskImage": {
                  "source": {
                    "id": "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccount'))]",
                    "uri": "[parameters('vhdBlobURL')]"
                  }
                }
              }
            }
          }
        ]
      },
      {
        "apiVersion": "2021-10-01",
        "type": "images",
        "name": "[variables('imageNameGen2')]",
        "location": "[variables('location')]",
        "dependsOn": [
          "[variables('galleryName')]"
        ],
        "properties": {
          "architecture": "[parameters('architecture')]",
          "hyperVGeneration": "V2",
          "identifier": {
            "offer": "rhcos-gen2",
            "publisher": "RedHat-gen2",
            "sku": "gen2"
          },
          "osState": "Generalized",
          "osType": "Linux"
```

```
        },
        "resources": [
          {
            "apiVersion": "2021-10-01",
            "type": "versions",
            "name": "[variables('imageRelease')]",
            "location": "[variables('location')]",
            "dependsOn": [
              "[variables('imageNameGen2')]"
            ],
            "properties": {
              "publishingProfile": {
                "storageAccountType": "Standard_LRS",
                "targetRegions": [
                  {
                    "name": "[variables('location')]",
                    "regionalReplicaCount": "1"
                  }
                ]
              },
              "storageProfile": {
                "osDiskImage": {
                  "source": {
                    "id": "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccount'))]",
                    "uri": "[parameters('vhdBlobURL')]"
                  }
                }
              }
            }
          }
        ]
      }
    ]
  }
}
```

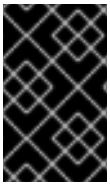
7.11.14. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

7.11.14.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表7.27 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表7.28 すべてのマシンからコントロールプレーンへの通信に使用されるポート

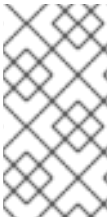
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表7.29 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

7.11.15. Azure でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用するネットワークおよび負荷分散を Microsoft Azure で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

提供される ARM テンプレートを使用して Azure インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。

手順

1. 本トピックの **ネットワークおよびロードばランサーの ARM テンプレート**セクションからテンプレートをコピーし、これを **03_infra.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。

2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/03_infra.json" \
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

1. プライベート DNS ゾーンの名前。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. API パブリックロードバランサーのパブリックゾーンに **api** DNS レコードを作成します。**${BASE_DOMAIN_RESOURCE_GROUP}** 変数は、パブリック DNS ゾーンがあるリソースグループをポイントする必要があります。

   a. 以下の変数をエクスポートします。

```
$ export PUBLIC_IP=`az network public-ip list -g ${RESOURCE_GROUP} --query "[?
name=='${INFRA_ID}-master-pip'] | [0].ipAddress" -o tsv`
```

b. 新しいパブリックゾーンに **api** DNS レコードを作成します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n api -a ${PUBLIC_IP} --ttl 60
```

クラスターを既存のパブリックゾーンに追加する場合は、**api** DNS レコードを代わりに作成できます。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n api.${CLUSTER_NAME} -a ${PUBLIC_IP} --ttl 60
```

7.11.15.1. ネットワークおよびロードバランサーの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用して、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例7.87 03_infra.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vnetBaseName": {
      "type": "string",
      "defaultValue": "",
      "metadata" : {
        "description" : "The specific customer vnet's base name (optional)"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
```

```
    "masterPublicIpAddressName" : "[concat(parameters('baseName'), '-master-pip')]",
    "masterPublicIpAddressID" : "[resourceId('Microsoft.Network/publicIPAddresses',
variables('masterPublicIpAddressName'))]",
    "masterLoadBalancerName" : "[parameters('baseName')]",
    "masterLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('masterLoadBalancerName'))]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "internalLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('internalLoadBalancerName'))]",
    "skuName": "Standard"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('masterPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('masterPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/loadBalancers",
      "name" : "[variables('masterLoadBalancerName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/publicIPAddresses/', variables('masterPublicIpAddressName'))]"
      ],
      "properties" : {
        "frontendIPConfigurations" : [
          {
            "name" : "public-lb-ip-v4",
            "properties" : {
              "publicIPAddress" : {
                "id" : "[variables('masterPublicIpAddressID')]"
              }
            }
          }
        ],
        "backendAddressPools" : [
          {
            "name" : "[variables('masterLoadBalancerName')]"
          }
        ],
        "loadBalancingRules" : [
          {
```

```
          "name" : "api-internal",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" :"[concat(variables('masterLoadBalancerID'), '/frontendIPConfigurations/public-lb-ip-
v4')]"
            },
            "backendAddressPool" : {
              "id" : "[concat(variables('masterLoadBalancerID'), '/backendAddressPools/',
variables('masterLoadBalancerName'))]"
            },
            "protocol" : "Tcp",
            "loadDistribution" : "Default",
            "idleTimeoutInMinutes" : 30,
            "frontendPort" : 6443,
            "backendPort" : 6443,
            "probe" : {
              "id" : "[concat(variables('masterLoadBalancerID'), '/probes/api-internal-probe')]"
            }
          }
        }
      ],
      "probes" : [
        {
          "name" : "api-internal-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 6443,
            "requestPath": "/readyz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        }
      ]
    }
  },
  {
    "apiVersion" : "2018-12-01",
    "type" : "Microsoft.Network/loadBalancers",
    "name" : "[variables('internalLoadBalancerName')]",
    "location" : "[variables('location')]",
    "sku": {
      "name": "[variables('skuName')]"
    },
    "properties" : {
      "frontendIPConfigurations" : [
        {
          "name" : "internal-lb-ip",
          "properties" : {
            "privateIPAllocationMethod" : "Dynamic",
            "subnet" : {
              "id" : "[variables('masterSubnetRef')]"
            },
            "privateIPAddressVersion" : "IPv4"
          }
        }
      ],
```

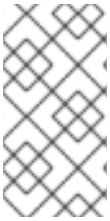
```
      "backendAddressPools" : [
        {
          "name" : "internal-lb-backend"
        }
      ],
      "loadBalancingRules" : [
        {
          "name" : "api-internal",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
            },
            "frontendPort" : 6443,
            "backendPort" : 6443,
            "enableFloatingIP" : false,
            "idleTimeoutInMinutes" : 30,
            "protocol" : "Tcp",
            "enableTcpReset" : false,
            "loadDistribution" : "Default",
            "backendAddressPool" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
            },
            "probe" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/probes/api-internal-probe')]"
            }
          }
        },
        {
          "name" : "sint",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
            },
            "frontendPort" : 22623,
            "backendPort" : 22623,
            "enableFloatingIP" : false,
            "idleTimeoutInMinutes" : 30,
            "protocol" : "Tcp",
            "enableTcpReset" : false,
            "loadDistribution" : "Default",
            "backendAddressPool" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
            },
            "probe" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/probes/sint-probe')]"
            }
          }
        }
      ],
      "probes" : [
        {
          "name" : "api-internal-probe",
```

```
            "properties" : {
              "protocol" : "Https",
              "port" : 6443,
              "requestPath": "/readyz",
              "intervalInSeconds" : 10,
              "numberOfProbes" : 3
            }
          },
          {
            "name" : "sint-probe",
            "properties" : {
              "protocol" : "Https",
              "port" : 22623,
              "requestPath": "/healthz",
              "intervalInSeconds" : 10,
              "numberOfProbes" : 3
            }
          }
        ]
      }
    },
    {
      "apiVersion": "2018-09-01",
      "type": "Microsoft.Network/privateDnsZones/A",
      "name": "[concat(parameters('privateDNSZoneName'), '/api')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
      ],
      "properties": {
        "ttl": 60,
        "aRecords": [
          {
            "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
          }
        ]
      }
    },
    {
      "apiVersion": "2018-09-01",
      "type": "Microsoft.Network/privateDnsZones/A",
      "name": "[concat(parameters('privateDNSZoneName'), '/api-int')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
      ],
      "properties": {
        "ttl": 60,
        "aRecords": [
          {
            "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
          }
```

```
        ]
      }
    }
  ]
}
```

7.11.16. Azure でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Microsoft Azure で作成する必要があります。このマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

提供されている ARM テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。

手順

1. 本トピックの **ブートストラップマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. ブートストラップ URL 変数をエクスポートします。

```
$ bootstrap_url_expiry=`date -u -d "10 hours" '+%Y-%m-%dT%H:%MZ'`
```

```
$ export BOOTSTRAP_URL=`az storage blob generate-sas -c 'files' -n 'bootstrap.ign' --https-only --full-uri --permissions r --expiry $bootstrap_url_expiry --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -o tsv`
```

3. ブートストラップ Ignition 変数をエクスポートします。

```
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "3.2.0" --arg url ${BOOTSTRAP_URL} '{ignition:{version:$v,config:{replace:{source:$url}}}}' | base64 | tr -d '\n'`
```

4. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/04_bootstrap.json" \
  --parameters bootstrapIgnition="${BOOTSTRAP_IGNITION}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameter bootstrapVMSize="Standard_D4s_v3" 3
```

1. ブートストラップクラスターのブートストラップ Ignition コンテンツ。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. オプション: ブートストラップ仮想マシンのサイズを指定します。指定したアーキテクチャーと互換性のある仮想マシンサイズを使用してください。この値が定義されていない場合は、テンプレートのデフォルト値が設定されます。

7.11.16.1. ブートストラップマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例7.88 04_bootstrap.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vnetBaseName": {
      "type": "string",
      "defaultValue": "",
      "metadata" : {
        "description" : "The specific customer vnet's base name (optional)"
      }
    },
    "bootstrapIgnition" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Bootstrap ignition content for the bootstrap cluster"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "defaultValue" : "Unused",
      "metadata" : {
        "description" : "Unused"
      }
```

```
    },
    "bootstrapVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "metadata" : {
        "description" : "The size of the Bootstrap Virtual Machine"
      }
    },
    "hyperVGen": {
      "type": "string",
      "metadata": {
        "description": "VM generation image to use"
      },
      "defaultValue": "V2",
      "allowedValues": [
        "V1",
        "V2"
      ]
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[parameters('baseName')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "vmName" : "[concat(parameters('baseName'), '-bootstrap')]",
    "nicName" : "[concat(variables('vmName'), '-nic')]",
    "galleryName": "[concat('gallery_', replace(parameters('baseName'), '-', '_'))]",
    "imageName" : "[concat(parameters('baseName'), if(equals(parameters('hyperVGen'), 'V2'), '-
gen2', ''))]",
    "clusterNsgName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-nsg')]",
    "sshPublicIpAddressName" : "[concat(variables('vmName'), '-ssh-pip')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('sshPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('sshPublicIpAddressName')]"
```

```
        }
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Network/networkInterfaces",
      "name" : "[variables('nicName')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[resourceId('Microsoft.Network/publicIPAddresses', variables('sshPublicIpAddressName'))]"
      ],
      "properties" : {
        "ipConfigurations" : [
          {
            "name" : "pipConfig",
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('sshPublicIpAddressName'))]"
              },
              "subnet" : {
                "id" : "[variables('masterSubnetRef')]"
              },
              "loadBalancerBackendAddressPools" : [
                {
                  "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/',
variables('masterLoadBalancerName'))]"
                },
                {
                  "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
                }
              ]
            }
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Compute/virtualMachines",
      "name" : "[variables('vmName')]",
      "location" : "[variables('location')]",
      "identity" : {
        "type" : "userAssigned",
        "userAssignedIdentities" : {
          "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
```

```
      ],
      "properties" : {
        "hardwareProfile" : {
          "vmSize" : "[parameters('bootstrapVMSize')]"
        },
        "osProfile" : {
          "computerName" : "[variables('vmName')]",
          "adminUsername" : "core",
          "adminPassword" : "NotActuallyApplied!",
          "customData" : "[parameters('bootstrapIgnition')]",
          "linuxConfiguration" : {
            "disablePasswordAuthentication" : false
          }
        },
        "storageProfile" : {
          "imageReference": {
            "id": "[resourceId('Microsoft.Compute/galleries/images', variables('galleryName'),
variables('imageName'))]"
          },
          "osDisk" : {
            "name": "[concat(variables('vmName'),'_OSDisk')]",
            "osType" : "Linux",
            "createOption" : "FromImage",
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB" : 100
          }
        },
        "networkProfile" : {
          "networkInterfaces" : [
            {
              "id" : "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"
            }
          ]
        }
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type": "Microsoft.Network/networkSecurityGroups/securityRules",
      "name" : "[concat(variables('clusterNsgName'), '/bootstrap_ssh_in')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[resourceId('Microsoft.Compute/virtualMachines', variables('vmName'))]"
      ],
      "properties": {
        "protocol" : "Tcp",
        "sourcePortRange" : "*",
        "destinationPortRange" : "22",
        "sourceAddressPrefix" : "*",
        "destinationAddressPrefix" : "*",
        "access" : "Allow",
        "priority" : 100,
        "direction" : "Inbound"
      }
```

```
    }
  ]
}
```

7.11.17. Azure でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Microsoft Azure で作成する必要があります。これらのマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

デフォルトでは、Microsoft Azure はコントロールプレーンマシンとコンピュートマシンを事前設定されたアベイラビリティーゾーンに配置します。コンピュートノードまたはコントロールプレーンノードのアベイラビリティーゾーンを手動で設定できます。これを行うには、仮想マシンリソースの **zones** パラメーターで各可用性ゾーンを指定して、ベンダーの Azure Resource Manager (ARM) テンプレートを変更します。

提供される ARM テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合は、インストールログで Red Hat サポートに接続することを検討してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの **コントロールプレーンマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **05_masters.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. コントロールプレーンマシンのデプロイメントに必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/05_masters.json" \
  --parameters masterIgnition="${MASTER_IGNITION}" \ 1
```

```
--parameters baseName="${INFRA_ID}" \ 2
--parameters masterVMSize="Standard_D8s_v3" 3
```

1 コントロールプレーンノードの Ignition コンテンツ。

2 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3 オプション: コントロールプレーン仮想マシンのサイズを指定します。指定したアーキテクチャーと互換性のある仮想マシンサイズを使用してください。この値が定義されていない場合は、テンプレートのデフォルト値が設定されます。

7.11.17.1. コントロールプレーンマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例7.89 05_masters.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vnetBaseName": {
      "type": "string",
      "defaultValue": "",
      "metadata" : {
        "description" : "The specific customer vnet's base name (optional)"
      }
    },
    "masterIgnition" : {
      "type" : "string",
      "metadata" : {
        "description" : "Ignition content for the master nodes"
      }
    },
    "numberOfMasters" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 2,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift masters to deploy"
      }
    },
    "sshKeyData" : {
```

```
      "type" : "securestring",
      "defaultValue" : "Unused",
      "metadata" : {
        "description" : "Unused"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "defaultValue" : "",
      "metadata" : {
        "description" : "unused"
      }
    },
    "masterVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D8s_v3",
      "metadata" : {
        "description" : "The size of the Master Virtual Machines"
      }
    },
    "diskSizeGB" : {
      "type" : "int",
      "defaultValue" : 1024,
      "metadata" : {
        "description" : "Size of the Master VM OS disk, in GB"
      }
    },
    "hyperVGen": {
      "type": "string",
      "metadata": {
        "description": "VM generation image to use"
      },
      "defaultValue": "V2",
      "allowedValues": [
        "V1",
        "V2"
      ]
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[parameters('baseName')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "galleryName": "[concat('gallery_', replace(parameters('baseName'), '-', '_'))]",
    "imageName" : "[concat(parameters('baseName'), if(equals(parameters('hyperVGen'), 'V2'), '-
gen2', ''))]",
```

```
    "copy" : [
     {
      "name" : "vmNames",
      "count" :  "[parameters('numberOfMasters')]",
      "input" : "[concat(parameters('baseName'), '-master-', copyIndex('vmNames'))]"
     }
    ]
   },
   "resources" : [
    {
     "apiVersion" : "2018-06-01",
     "type" : "Microsoft.Network/networkInterfaces",
     "copy" : {
      "name" : "nicCopy",
      "count" : "[length(variables('vmNames'))]"
     },
     "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
     "location" : "[variables('location')]",
     "properties" : {
      "ipConfigurations" : [
       {
        "name" : "pipConfig",
        "properties" : {
         "privateIPAllocationMethod" : "Dynamic",
         "subnet" : {
          "id" : "[variables('masterSubnetRef')]"
         },
         "loadBalancerBackendAddressPools" : [
          {
           "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/',
variables('masterLoadBalancerName'))]"
          },
          {
           "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
          }
         ]
        }
       }
      ]
     }
    },
    {
     "apiVersion" : "2018-06-01",
     "type" : "Microsoft.Compute/virtualMachines",
     "copy" : {
      "name" : "vmCopy",
      "count" : "[length(variables('vmNames'))]"
     },
     "name" : "[variables('vmNames')[copyIndex()]]",
     "location" : "[variables('location')]",
     "identity" : {
      "type" : "userAssigned",
```

```
        "userAssignedIdentities" : {
          "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]"
      ],
      "properties" : {
        "hardwareProfile" : {
          "vmSize" : "[parameters('masterVMSize')]"
        },
        "osProfile" : {
          "computerName" : "[variables('vmNames')[copyIndex()]]",
          "adminUsername" : "core",
          "adminPassword" : "NotActuallyApplied!",
          "customData" : "[parameters('masterIgnition')]",
          "linuxConfiguration" : {
            "disablePasswordAuthentication" : false
          }
        },
        "storageProfile" : {
          "imageReference": {
            "id": "[resourceId('Microsoft.Compute/galleries/images', variables('galleryName'),
variables('imageName'))]"
          },
          "osDisk" : {
            "name": "[concat(variables('vmNames')[copyIndex()], '_OSDisk')]",
            "osType" : "Linux",
            "createOption" : "FromImage",
            "caching": "ReadOnly",
            "writeAcceleratorEnabled": false,
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB" : "[parameters('diskSizeGB')]"
          }
        },
        "networkProfile" : {
          "networkInterfaces" : [
            {
              "id" : "[resourceId('Microsoft.Network/networkInterfaces', concat(variables('vmNames')
[copyIndex()], '-nic'))]",
              "properties": {
                "primary": false
              }
            }
          ]
        }
      }
    }
  ]
}
```

7.11.18. ブートストラップの完了を待機し、Azure のブートストラップリソースを削除する

Microsoft Azure ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ az network nsg rule delete -g ${RESOURCE_GROUP} --nsg-name ${INFRA_ID}-nsg --
name bootstrap_ssh_in
$ az vm stop -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm deallocate -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap --yes
$ az disk delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap_OSDisk --no-
wait --yes
$ az network nic delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-nic --no-
wait
$ az storage blob delete --account-key ${ACCOUNT_KEY} --account-name
${CLUSTER_NAME}sa --container-name files --name bootstrap.ign
$ az network public-ip delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-
ssh-pip
```

-

注記

ブートストラップサーバーを削除しないと、API トラフィックがブートストラップサーバーにルーティングされるため、インストールが成功しない場合があります。

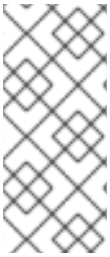
7.11.19. Azure での追加のワーカーマシンの作成

Microsoft Azure でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

注記

3 ノードクラスターをインストールする場合は、この手順をスキップしてください。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコントロールプレーンマシンで設定されます。

この例では、Azure Resource Manager (ARM) テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_workers.json** というタイプのリソースを追加して起動することができます。



注記

デフォルトでは、Microsoft Azure はコントロールプレーンマシンとコンピュートマシンを事前設定されたアベイラビリティーゾーンに配置します。コンピュートノードまたはコントロールプレーンノードのアベイラビリティーゾーンを手動で設定できます。これを行うには、仮想マシンリソースの **zones** パラメーターに各アベイラビリティーゾーンを指定して、ベンダーの ARM テンプレートを変更します。

提供される ARM テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合は、インストールログで Red Hat サポートに接続することを検討してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの **ワーカーマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **06_workers.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。

2. ワーカーマシンのデプロイメントで必要な以下の変数をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/06_workers.json" \
  --parameters workerIgnition="${WORKER_IGNITION}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameters nodeVMSize="Standard_D4s_v3" 3
```

1. ワーカーノードの Ignition コンテンツ。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. オプション: コンピュートノード仮想マシンのサイズを指定します。指定したアーキテクチャーと互換性のある仮想マシンサイズを使用してください。この値が定義されていない場合は、テンプレートのデフォルト値が設定されます。

7.11.19.1. ワーカーマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例7.90 06_workers.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vnetBaseName": {
      "type": "string",
      "defaultValue": "",
      "metadata" : {
        "description" : "The specific customer vnet's base name (optional)"
      }
    },
    "workerIgnition" : {
      "type" : "string",
      "metadata" : {
```

```
        "description" : "Ignition content for the worker nodes"
      }
    },
    "numberOfNodes" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 2,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift compute nodes to deploy"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "defaultValue" : "Unused",
      "metadata" : {
        "description" : "Unused"
      }
    },
    "nodeVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "metadata" : {
        "description" : "The size of the each Node Virtual Machine"
      }
    },
    "hyperVGen": {
      "type": "string",
      "metadata": {
        "description": "VM generation image to use"
      },
      "defaultValue": "V2",
      "allowedValues": [
        "V1",
        "V2"
      ]
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "nodeSubnetName" : "[concat(if(not(empty(parameters('vnetBaseName'))),
parameters('vnetBaseName'), parameters('baseName')), '-worker-subnet')]",
    "nodeSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('nodeSubnetName'))]",
    "infraLoadBalancerName" : "[parameters('baseName')]",
    "sshKeyPath" : "/home/capi/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "galleryName": "[concat('gallery_', replace(parameters('baseName'), '-', '_'))]",
    "imageName" : "[concat(parameters('baseName'), if(equals(parameters('hyperVGen'), 'V2'), '-
gen2', ''))]",
    "copy" : [
      {
```

```
        "name" : "vmNames",
        "count" :  "[parameters('numberOfNodes')]",
        "input" : "[concat(parameters('baseName'), '-worker-', variables('location'), '-',
copyIndex('vmNames', 1))]"
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2019-05-01",
      "name" : "[concat('node', copyIndex())]",
      "type" : "Microsoft.Resources/deployments",
      "copy" : {
        "name" : "nodeCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "properties" : {
        "mode" : "Incremental",
        "template" : {
          "$schema" : "http://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
          "contentVersion" : "1.0.0.0",
          "resources" : [
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Network/networkInterfaces",
              "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
              "location" : "[variables('location')]",
              "properties" : {
                "ipConfigurations" : [
                  {
                    "name" : "pipConfig",
                    "properties" : {
                      "privateIPAllocationMethod" : "Dynamic",
                      "subnet" : {
                        "id" : "[variables('nodeSubnetRef')]"
                      }
                    }
                  }
                ]
              }
            },
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Compute/virtualMachines",
              "name" : "[variables('vmNames')[copyIndex()]]",
              "location" : "[variables('location')]",
              "tags" : {
                "kubernetes.io-cluster-ffranzupi": "owned"
              },
              "identity" : {
                "type" : "userAssigned",
                "userAssignedIdentities" : {
                  "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
                }
```

```
            },
            "dependsOn" : [
              "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')
[copyIndex()], '-nic'))]"
            ],
            "properties" : {
              "hardwareProfile" : {
                "vmSize" : "[parameters('nodeVMSize')]"
              },
              "osProfile" : {
                "computerName" : "[variables('vmNames')[copyIndex()]]",
                "adminUsername" : "capi",
                "adminPassword" : "NotActuallyApplied!",
                "customData" : "[parameters('workerIgnition')]",
                "linuxConfiguration" : {
                  "disablePasswordAuthentication" : false
                }
              },
              "storageProfile" : {
                "imageReference": {
                  "id": "[resourceId('Microsoft.Compute/galleries/images', variables('galleryName'),
variables('imageName'))]"
                },
                "osDisk" : {
                  "name": "[concat(variables('vmNames')[copyIndex()],'_OSDisk')]",
                  "osType" : "Linux",
                  "createOption" : "FromImage",
                  "managedDisk": {
                    "storageAccountType": "Premium_LRS"
                  },
                  "diskSizeGB": 128
                }
              },
              "networkProfile" : {
                "networkInterfaces" : [
                  {
                    "id" : "[resourceId('Microsoft.Network/networkInterfaces',
concat(variables('vmNames')[copyIndex()], '-nic'))]",
                    "properties": {
                      "primary": true
                    }
                  }
                ]
              }
            }
          }
        ]
      }
    }
  ]
}
```

7.11.20. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。

5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.11.21. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。

- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

7.11.22. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
master-2 Ready    master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

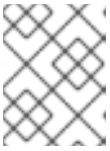
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

7.11.23. Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用して、OpenShift Container Platform クラスターを Microsoft Azure にデプロイしています。
- OpenShift CLI (**oc**) をインストールすること。
- Azure CLI のインストールまたは更新を実行します。

手順

1. Ingress ルーターがロードバランサーを作成し、**EXTERNAL-IP** フィールドにデータを設定していることを確認します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.20.10  35.130.120.110
80:32288/TCP,443:31215/TCP  20
```

2. Ingress ルーター IP を変数としてエクスポートします。

```
$ export PUBLIC_IP_ROUTER=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

3. パブリック DNS ゾーンに ***.apps** レコードを追加します。

   a. このクラスターを新しいパブリックゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER} --ttl 300
```

   b. このクラスターを既存のパブリックゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n *.apps.${CLUSTER_NAME} -a ${PUBLIC_IP_ROUTER} --ttl 300
```

4. ***.apps** レコードをプライベート DNS ゾーンに追加します。

   a. 以下のコマンドを使用して ***.apps** レコードを作成します。

   ```
   $ az network private-dns record-set a create -g ${RESOURCE_GROUP} -z
   ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps --ttl 300
   ```

   b. 以下のコマンドを使用して ***.apps** レコードをプライベート DNS ゾーンに追加します。

   ```
   $ az network private-dns record-set a add-record -g ${RESOURCE_GROUP} -z
   ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER}
   ```

ワイルドカードを使用する代わりに明示的なドメインを追加する場合は、クラスターのそれぞれの現行ルートのエントリーを作成できます。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}
{end}' routes
```

出力例

```
oauth-openshift.apps.cluster.basedomain.com
console-openshift-console.apps.cluster.basedomain.com
downloads-openshift-console.apps.cluster.basedomain.com
alertmanager-main-openshift-monitoring.apps.cluster.basedomain.com
prometheus-k8s-openshift-monitoring.apps.cluster.basedomain.com
```

7.11.24. ユーザーによってプロビジョニングされるインフラストラクチャーでの Azure インストールの実行

Microsoft Azure のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる Azure インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

- クラスターのインストールを完了します。

  ```
  $ ./openshift-install --dir <installation_directory> wait-for install-complete 1
  ```

  出力例

  ```
  INFO Waiting up to 30m0s for the cluster to initialize...
  ```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

7.11.25. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.12. 制限されたネットワーク内の AZURE にクラスターをインストールする

OpenShift Container Platform バージョン 4.16 では、既存の Azure Virtual Network (VNet) 上にインストールリリースコンテンツの内部ミラーを作成することで、制限されたネットワーク内の Microsoft Azure にクラスターをインストールできます。

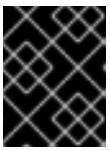
重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが Azure API を使用するにはインターネットアクセスが必要になります。

7.12.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。

- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- Azure に既存の VNet がある。インストーラーでプロビジョニングされるインフラストラクチャーを使用するネットワークが制限された環境にクラスターをインストールする場合は、インストーラーでプロビジョニングされる VNet を使用することはできません。以下の要件のいずれかを満たすユーザーによってプロビジョニングされる VPC を使用する必要があります。
  - VNet にはミラーレジストリーが含まれています
  - VNet に別の場所でホストされるミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- 暗号化のために Azure 環境を準備 した (顧客管理の暗号化キーを使用する場合)。

7.12.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

7.12.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

7.12.2.2. ユーザー定義のアウトバウンドルーティング

OpenShift Container Platform では、クラスターがインターネットに接続するために独自のアウトバウンドルーティングを選択できます。これにより、パブリック IP アドレスおよびパブリックロードバランサーの作成を省略できます。

クラスターをインストールする前に、**install-config.yaml** ファイルのパラメーターを変更してユーザー定義のルーティングを設定できます。クラスターのインストール時にアウトバウンドルーティングを使用するには、既存の VNet が必要です。インストールプログラムはこれを設定しません。

クラスターをユーザー定義のルーティングを使用するように設定する際に、インストールプログラムは以下のリソースを作成しません。

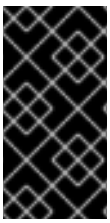
- インターネットにアクセスするためのアウトバウンドルール。
- パブリックロードバランサーのパブリック IP。
- アウトバウンド要求のパブリックロードバランサーにクラスターマシンを追加する Kubernetes Service オブジェクト。

ユーザー定義のルーティングを設定する前に、以下の項目が利用可能であることを確認する必要があります。

- OpenShift イメージレジストリーミラーを使用しない場合は、コンテナーイメージのプルにインターネットへの egress を使用できます。
- クラスターは Azure API にアクセスできます。
- 各種の allowlist エンドポイントが設定されます。これらのエンドポイントについては、**ファイアウォールの設定**セクションで参照できます。

ユーザー定義のルーティングを使用したインターネットアクセスでサポートされる既存のネットワーク設定がいくつかあります。

Azure Firewall を使用した制限付きクラスター
Azure Firewall を使用して、OpenShift Container Platform クラスターのインストールに使用される仮想ネットワーク (VNet) のアウトバウンドルーティングを制限できます。詳細は、Azure Firewall でのユーザー定義ルーティングの提供 を参照してください。Azure Firewall で VNet を使用し、ユーザー定義のルーティングを設定することで、制限されたネットワークに OpenShift Container Platform クラスターを作成できます。



重要

インターネットアクセスの制限に Azure Firewall を使用している場合は、**install-config.yaml** ファイルで **publish** フィールドを **Internal** に設定する必要があります。これは、Azure Firewall が Azure パブリックロードバランサーと正しく動作しない ためです。

7.12.3. OpenShift Container Platform クラスターでの VNet の再利用について

OpenShift Container Platform 4.16 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

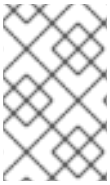
OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可

能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

7.12.3.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加のネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

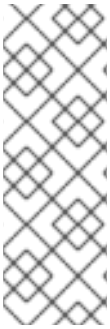
カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。



注記

デフォルトでは、**install-config.yaml** ファイルでアベイラビリティゾーンを指定すると、インストールプログラムはコントロールプレーンマシンとコンピューティングマシンを リージョン 内の これらのアベイラビリティゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

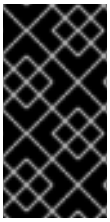
- 指定されたサブネットがすべて存在します。
- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットがあります
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。必要な場合に、インストールプログラムはコントロールプレーンおよびワーカーノードを管理するパブリックロードバランサーを作成し、Azure はパブリック IP アドレスをそれらに割り当てます。

注記

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

7.12.3.1.1. ネットワークセキュリティーグループの要件

コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。



重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表7.30 必須ポート

ポート	説明	コントロールプレーン	Compute
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	

ポート	説明	コントロールプレーン	Compute
22623	マシンをプロビジョニングするためのマシン設定サーバーへの内部通信を許可します。	x	
*	Azure API への接続を許可します。宛先サービスタグを **AzureCloud** に設定する必要があります。[1]	x	x
*	インターネットへの接続を拒否します。宛先サービスタグを **Internet** に設定する必要があります。[1]	x	x

1. Azure Firewall を使用してインターネットアクセスを制限している場合は、Azure API を許可するように Azure Firewall を設定できます。ネットワークセキュリティーグループルールは必要ありません。

重要

現在、マシン設定サーバーエンドポイントをブロックまたは制限する方法はサポートされていません。マシン設定サーバーは、既存の設定または状態を持たない新しくプロビジョニングされたマシンが設定を取得できるように、ネットワークに公開する必要があります。このモデルでは、信頼のルートは証明書署名要求 (CSR) エンドポイントであり、kubelet がクラスターに参加するための承認のために証明書署名要求を送信する場所です。このため、シークレットや証明書などの機密情報を配布するためにマシン設定を使用しないでください。

マシン設定サーバーエンドポイント、ポート 22623 および 22624 がベアメタルシナリオで確実に保護されるようにするには、顧客は適切なネットワークポリシーを設定する必要があります。

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

関連情報

- OpenShift SDN ネットワークプラグインについて
- ファイアウォールの設定

7.12.3.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテー

ブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

7.12.3.3. クラスター間の分離

クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

7.12.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

7.12.5. クラスターノードの SSH アクセス用のキーペアの生成

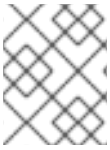
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/.ssh/id_ed25519 などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.12.6. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、アクセス可能な場所にアップロードしました。
- Azure サブスクリプション ID とテナント ID がある。
- サービスプリンシパルを使用してクラスターをインストールしている場合は、そのアプリケーション ID とパスワードが必要です。
- システムが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、インストールプログラムを実行する仮想マシン上でそれが有効になっています。
- ユーザーが割り当てたマネージド ID を使用してクラスターをインストールしている場合は、次の前提条件を満たしている必要があります。
  - そのクライアント ID がある。
  - これは、インストールプログラムを実行する仮想マシンに割り当てられている。

手順

1. オプション: 以前にこのコンピューターでインストールプログラムを実行したことがあり、代替のサービスプリンシパルまたはマネージド ID を使用する場合は、**~/.azure/** ディレクトリーに移動して、**osServicePrincipal.json** 設定ファイルを削除します。
   このファイルを削除すると、インストールプログラムが以前のインストールのサブスクリプション値と認証値を自動的に再利用できなくなります。

2. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **azure** を選択します。
      インストールプログラムが以前のインストールの **osServicePrincipal.json** 設定ファイルを見つけることができない場合は、Azure サブスクリプションと認証の値の入力を求められます。

      iii. サブスクリプションの次の Azure パラメーター値を入力します。

      - **azure subscription id** クラスターに使用するサブスクリプション ID を入力します。
      - **azure tenant id** テナント ID を入力します。

      iv. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアント ID** の入力を求められたら、次のいずれかを行います。

      - サービスプリンシパルを使用している場合は、そのアプリケーション ID を入力します。
      - システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
      - ユーザー割り当てのマネージド ID を使用している場合は、そのクライアント ID を指定します。

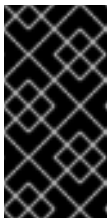
v. クラスターのデプロイに使用している Azure ID に応じて、**azure サービスプリンシパルのクライアントシークレット** の入力を求められたら、次のいずれかを実行します。

- サービスプリンシパルを使用している場合は、そのパスワードを入力します。
- システム割り当てのマネージド ID を使用している場合は、この値を空白のままにします。
- ユーザー割り当てのマネージド ID を使用している場合は、この値を空白のままにします。

vi. クラスターをデプロイするリージョンを選択します。

vii. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

viii. クラスターの記述名を入力します。



重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

ix. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
 -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
 -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. **platform.azure** フィールドでクラスターをインストールするための VNet のネットワークとサブネットを定義します。

```
networkResourceGroupName: <vnet_resource_group> 1
```

```
virtualNetwork: <vnet> 2
controlPlaneSubnet: <control_plane_subnet> 3
computeSubnet: <compute_subnet> 4
```

1. **<vnet_resource_group>** を、既存の仮想ネットワーク (VNet) を含むリソースグループ名に置き換えます。
2. **<vnet>** を既存の仮想ネットワーク名に置き換えます。
3. **<control_plane_subnet>** は、コントロールプレーンマシンをデプロイする既存のサブネット名に置き換えます。
4. **<compute_subnet>** を、コンピュートマシンをデプロイするための既存のサブネット名に置き換えます。

d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

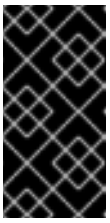
```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

e. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

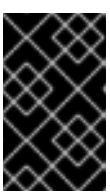


重要

Azure Firewall は、Azure Public ロードバランサーと シームレスに連携しません。したがって、インターネットアクセスを制限するために Azure Firewall を使用する場合は、**install-config.yaml** の **publish** フィールドを **Internal** に設定する必要があります。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

以前に検出されなかった場合は、インストールプログラムが **osServicePrincipal.json** 設定ファイルを作成し、このファイルをコンピューターの **~/.azure/** ディレクトリーに保存します。これにより、インストールプログラムがターゲットプラットフォーム上で OpenShift Container Platform クラスターを作成するときにプロファイルをロードできるようになります。

関連情報

- Azure のインストール設定パラメーター

7.12.6.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

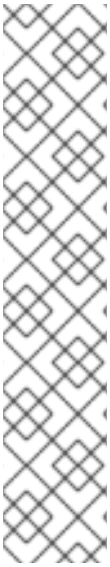
表7.31 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

重要

premiumIO パラメーターが **true** に設定されている Azure 仮想マシンを使用する必要があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

7.12.6.2. Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure インスタンスタイプは OpenShift Container Platform でテストされています。

例7.91 64 ビット x86 アーキテクチャーに基づくマシンタイプ

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***

- **t3.***
- **t3a.***

7.12.6.3. 64 ビット ARM インフラストラクチャー上の Azure のテスト済みインスタンスタイプ

以下の Microsoft Azure Azure64 インスタンスタイプは OpenShift Container Platform でテストされています。

例7.92 64 ビット ARM アーキテクチャーに基づくマシンタイプ

- **c6g.***
- **m6g.***

7.12.6.4. Azure VM の信頼された起動の有効化

Azure にクラスターをインストールするときに、セキュアブート と 仮想化された信頼できるプラットフォームモジュール という 2 つの信頼された起動機能を有効にできます。

これらの機能をサポートする 仮想マシンのサイズ については、仮想マシンのサイズに関する Azure のドキュメントを参照してください。

重要

信頼できる起動はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

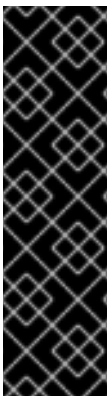
```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: TrustedLaunch 2
        trustedLaunch:
```

```
    uefiSettings:
      secureBoot: Enabled 3
      virtualizedTrustedPlatformModule: Enabled 4
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定すると、それぞれコントロールプレーンまたはコンピュートノードのみで信頼された起動が有効になります。すべてのノードで信頼できる起動を有効にするには、**platform.azure.defaultMachinePlatform** を指定します。
2. 信頼できる起動機能を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。

7.12.6.5. Confidential VM の有効化

クラスターをインストールするときに、Confidential 仮想マシンを有効にできます。コンピューティングノード、コンピュートノード、またはすべてのノードに対して Confidential 仮想マシンを有効にできます。



重要

Confidential VMs の使用はテクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

次の仮想マシンサイズの Confidential VM を使用できます。

- DCasv5 シリーズ
- DCadsv5 シリーズ
- ECasv5 シリーズ
- ECadsv5 シリーズ

重要

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

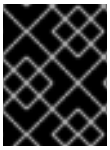
- クラスターをデプロイする前に、テキストエディターを使用して **install-config.yaml** ファイルを編集し、次のスタンザを追加します。

```
controlPlane: 1
  platform:
    azure:
      settings:
        securityType: ConfidentialVM 2
        confidentialVM:
          uefiSettings:
            secureBoot: Enabled 3
            virtualizedTrustedPlatformModule: Enabled 4
      osDisk:
        securityProfile:
          securityEncryptionType: VMGuestStateOnly 5
```

1. **controlPlane.platform.azure** または **compute.platform.azure** を指定して、それぞれコントロールプレーンまたはコンピュートノードのみに Confidential 仮想マシンをデプロイします。すべてのノードに Confidential 仮想マシンをデプロイするには、**platform.azure.defaultMachinePlatform** を指定します。
2. Confidential VM を有効にします。
3. Secure Boot を有効にします。詳細は、セキュアブート に関する Azure ドキュメントを参照してください。
4. 仮想化された Trusted Platform Module を有効にします。詳細は、仮想化された Trusted Platform Module に関する Azure のドキュメントを参照してください。
5. 仮想マシンゲストの状態を暗号化するには、**VMGuestStateOnly** を指定します。

7.12.6.6. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    azure:
      encryptionAtHost: true
      ultraSSDCapability: Enabled
      osDisk:
```

```
        diskSizeGB: 1024 5
        diskType: Premium_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      type: Standard_D8s_v3
  replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      ultraSSDCapability: Enabled
      type: Standard_D2s_v3
      encryptionAtHost: true
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
        diskEncryptionSet:
          resourceGroup: disk_encryption_set_resource_group
          name: disk_encryption_set_name
          subscriptionId: secondary_subscription_id
      osImage:
        publisher: example_publisher_name
        offer: example_image_offer
        sku: example_offer_sku
        version: example_image_version
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 11
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    defaultMachinePlatform:
      osImage: 12
        publisher: example_publisher_name
        offer: example_image_offer
```

```
        sku: example_offer_sku
        version: example_image_version
      ultraSSDCapability: Enabled
    baseDomainResourceGroupName: resource_group 13
    region: centralus 14
    resourceGroupName: existing_resource_group 15
    networkResourceGroupName: vnet_resource_group 16
    virtualNetwork: vnet 17
    controlPlaneSubnet: control_plane_subnet 18
    computeSubnet: compute_subnet 19
    outboundType: UserDefinedRouting 20
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 21
fips: false 22
sshKey: ssh-ed25519 AAAA... 23
additionalTrustBundle: | 24
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: 25
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
publish: Internal 26
```

1 10 14 21 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンのリストを指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 オプション: コントロールプレーンとコンピュートマシンを起動するために使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) イメージ。**platform.azure.defaultMachinePlatform.osImage** の下の **publisher**、**offer**、**sku**、および **version** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**controlPlane.platform.azure.osImage** または **compute.platform.azure.osImage** の下のパラメーターが設定されている場合、それらは **platform.azure.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

13 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

15 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

16 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

17 既存の VNet を使用する場合は、その名前を指定します。

18 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

19 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

20 Azure Firewall を使用してインターネットアクセスを制限する場合は、Azure Firewall 経由でトラフィックを送信するように送信ルーティングを設定する必要があります。ユーザー定義のルーティングを設定すると、クラスターに外部エンドポイントが公開されなくなります。

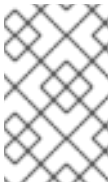
22 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

23 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

24 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

25 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

26 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。Azure Firewall を使用してインターネットアクセスを制限する場合は、**publish** を **Internal** に設定してプライベートクラスターをデプロイします。ユーザー側のエンドポイントにはインターネットからアクセスできなくなります。デフォルト値は **External** です。

7.12.6.7. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

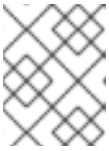
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
```

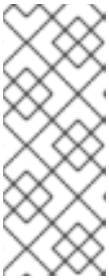
```
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

7.12.7. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

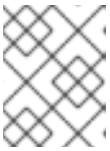
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.12.8. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように Azure クラスターを設定する の手順に従ってください。

7.12.8.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

3. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
    ...
```

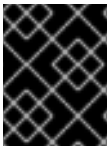
5. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
    ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

7.12.8.2. 短期認証情報を使用するように Azure クラスターを設定する

Microsoft Entra Workload ID を使用するクラスターをインストールするには、Cloud Credential Operator ユーティリティーを設定し、クラスターに必要な Azure リソースを作成する必要があります。

7.12.8.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- 次の権限で使用する **ccoctl** ユーティリティー用のグローバル Microsoft Azure アカウントが作成されました。

  例7.93 必要な Azure 権限

  - Microsoft.Resources/subscriptions/resourceGroups/read
  - Microsoft.Resources/subscriptions/resourceGroups/write
  - Microsoft.Resources/subscriptions/resourceGroups/delete
  - Microsoft.Authorization/roleAssignments/read
  - Microsoft.Authorization/roleAssignments/delete

- Microsoft.Authorization/roleAssignments/write
- Microsoft.Authorization/roleDefinitions/read
- Microsoft.Authorization/roleDefinitions/write
- Microsoft.Authorization/roleDefinitions/delete
- Microsoft.Storage/storageAccounts/listkeys/action
- Microsoft.Storage/storageAccounts/delete
- Microsoft.Storage/storageAccounts/read
- Microsoft.Storage/storageAccounts/write
- Microsoft.Storage/storageAccounts/blobServices/containers/write
- Microsoft.Storage/storageAccounts/blobServices/containers/delete
- Microsoft.Storage/storageAccounts/blobServices/containers/read
- Microsoft.ManagedIdentity/userAssignedIdentities/delete
- Microsoft.ManagedIdentity/userAssignedIdentities/read
- Microsoft.ManagedIdentity/userAssignedIdentities/write
- Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/read
- Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/write
- Microsoft.ManagedIdentity/userAssignedIdentities/federatedIdentityCredentials/delete
- Microsoft.Storage/register/action
- Microsoft.ManagedIdentity/register/action

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

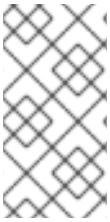
```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

7.12.8.2.2. Cloud Credential Operator ユーティリティーを使用した Azure リソースの作成

ccoctl azure create-all コマンドを使用して、Azure リソースの作成を自動化できます。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。
- Azure CLI を使用して Microsoft Azure アカウントにアクセスします。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. **ccoctl** ユーティリティーが Azure 認証情報を自動的に検出できるようにするには、次のコマンドを実行して Azure CLI にログインします。

```
$ az login
```

4. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl azure create-all \
```

```
--name=<azure_infra_name> \1
--output-dir=<ccoctl_output_dir> \2
--region=<azure_region> \3
--subscription-id=<azure_subscription_id> \4
--credentials-requests-dir=<path_to_credentials_requests_directory> \5
--dnszone-resource-group-name=<azure_dns_zone_resource_group_name> \6
--tenant-id=<azure_tenant_id> 7
```

1. トラッキングに使用される、作成されたすべての Azure リソースのユーザー定義名を指定します。
2. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
3. クラウドリソースが作成される Azure リージョンです。
4. 使用する Azure サブスクリプション ID を指定します。
5. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
6. クラスターのベースドメイン Azure DNS ゾーンを含むリソースグループの名前を指定します。
7. 使用する Azure テナント ID を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

追加のオプションパラメーターとその使用方法の説明を表示するには、**azure create-all --help** コマンドを実行します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
azure-ad-pod-identity-webhook-config.yaml
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-azure-cloud-credentials-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capz-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-disk-credentials-credentials.yaml
openshift-cluster-csi-drivers-azure-file-credentials-credentials.yaml
```

```
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-azure-cloud-credentials-credentials.yaml
```

Azure をクエリーすることで、Microsoft Entra ID サービスアカウントが作成されていることを確認できます。詳細は、Entra ID サービスアカウントのリストに関する Azure ドキュメントを参照してください。

7.12.8.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. 既存のリソースグループを使用する代わりに、**ccoctl** ユーティリティーを使用して新しい Azure リソースグループを作成した場合は、次のように **install-config.yaml** の **resourceGroupName** パラメーターを変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   # ...
   platform:
     azure:
       resourceGroupName: <azure_infra_name> 1
   # ...
   ```

   1 この値は、**ccoctl azure create-all** コマンドの **--name** 引数で指定された Azure リソースのユーザー定義名と一致する必要があります。

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

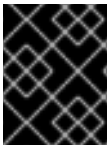
```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

7.12.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Azure サブスクリプション ID とテナント ID がある。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

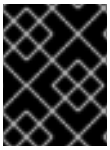
```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

7.12.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

7.12.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

7.12.12. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

7.13. AZURE に 3 ノードクラスターをインストールする

OpenShift Container Platform バージョン 4.16 では、Microsoft Azure に 3 ノードクラスターをインストールできます。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコントロールプレーンマシンで設定されます。このタイプのクラスターは、クラスター管理者および開発者がテスト、開発、および実稼働に使用するためのより小さくリソース効率の高いクラスターを提供します。

インストーラーによってプロビジョニングされたインフラストラクチャーまたはユーザーによってプロビジョニングされたインフラストラクチャーのいずれかを使用して、3 ノードクラスターをインストールできます。

注記

Azure Marketplace イメージを使用した 3 ノードクラスターのデプロイはサポートされていません。

7.13.1. 3 ノードクラスターの設定

クラスターをデプロイする前に、**install-config.yaml** ファイルでワーカーノードの数を **0** に設定して、3 ノードクラスターを設定します。ワーカーノードの数を **0** に設定すると、コントロールプレーンマシンがスケジュール可能になります。これにより、アプリケーションワークロードをコントロールプレーンノードから実行するようにスケジュールできます。

注記

アプリケーションワークロードはコントロールプレーンノードから実行され、コントロールプレーンノードはコンピュートノードと見なされるため、追加のサブスクリプションが必要です。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

1. 次の **compute** スタンザに示すように、**install-config.yaml** ファイルでコンピューティングレプリカの数を **0** に設定します。

   3 ノードクラスターの install-config.yaml ファイルの例

   ```
   apiVersion: v1
   baseDomain: example.com
   compute:
   - name: worker
     platform: {}
     replicas: 0
   # ...
   ```

2. ユーザーがプロビジョニングしたインフラストラクチャーを使用して、クラスターをデプロイする場合:

   - Kubernetes マニフェストファイルを作成したら、**cluster-scheduler-02-config.yml** ファイルで **spec.mastersSchedulable** パラメーターが **true** に設定されていることを確認します。このファイルは、**<installation_directory>/manifests** にあります。詳細については、「ARM テンプレートを使用して、Azure にクラスターをインストールする」の「Kubernetes マニフェストと Ignition 設定ファイルの作成」を参照してください。

   - 追加のワーカーノードを作成しないでください。

3 ノードクラスターの cluster-scheduler-02-config.yml ファイルの例

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  creationTimestamp: null
  name: cluster
spec:
  mastersSchedulable: true
  policy:
    name: ""
status: {}
```

-

7.13.2. 次のステップ

- カスタマイズによる Azure へのクラスターのインストール
- ARM テンプレートを使用したクラスターの Azure へのインストール

7.14. AZURE でのクラスターのアンインストール

Microsoft Azure にデプロイしたクラスターは削除することができます。

7.14.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

7.14.2. Cloud Credential Operator ユーティリティーを使用した Microsoft Azure リソースの削除

クラスターの外部で管理される短期認証情報を使用する OpenShift Container Platform クラスターをアンインストールした後、CCO ユーティリティー (**ccoctl**) を使用して、インストール中に **ccoctl** が作成した Microsoft Azure (Azure) リソースを削除できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。
- 短期認証情報を使用する Azure 上の OpenShift Container Platform クラスターをアンインストールします。

手順

- 次のコマンドを実行して、**ccoctl** が作成した Azure リソースを削除します。

```
$ ccoctl azure delete \
  --name=<name> \ 1
  --region=<azure_region> \ 2
  --subscription-id=<azure_subscription_id> \ 3
  --delete-oidc-resource-group
```

1. **<name>** は、クラウドリソースを最初に作成してタグ付けするために使用された名前と一致します。
2. **<azure_region>** は、クラウドリソースを削除する Azure リージョンです。
3. **<azure_subscription_id>** は、クラウドリソースを削除する Azure サブスクリプション ID です。

検証

- リソースが削除されたことを確認するには、Azure にクエリーを実行します。詳細は、Azure のドキュメントを参照してください。

7.15. AZURE のインストール設定パラメーター

OpenShift Container Platform クラスターを Microsoft Azure にデプロイする前に、パラメーターを指定してクラスターとそれをホストするプラットフォームをカスタマイズします。**install-config.yaml** ファイルを作成するときは、コマンドラインを使用して必要なパラメーターの値を指定します。その後、**install-config.yaml** ファイルを変更して、クラスターをさらにカスタマイズできます。

7.15.1. Azure で使用可能なインストール設定パラメーター

次の表では、インストールプロセスの一部として設定できる、必須、オプション、および Azure 固有のインストール設定パラメーターを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

7.15.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表7.32 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object

パラメーター	説明	値
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

7.15.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

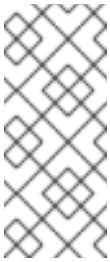
注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表7.33 ネットワークパラメーター

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

7.15.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

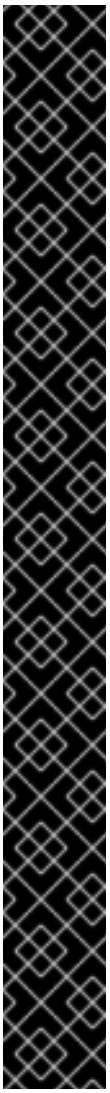
表7.34 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列

パラメーター	説明	値
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** と **arm64** です。すべてのインストールオプションが 64 ビット ARM アーキテクチャーをサポートしているわけではありません。使用するインストールオプションがプラットフォームでサポートされているか確認するには、**クラスターインストール方法の選択およびそのユーザー向けの準備** の **各種プラットフォームでサポートされているインストール方法** 参照してください。	String

パラメーター	説明	値
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** と **arm64** です。すべてのインストールオプションが 64 ビット ARM アーキテクチャーをサポートしているわけではありません。使用するインストールオプションがプラットフォームでサポートされているか確認するには、**クラスターインストール方法の選択およびそのユーザー向けの準備** の **各種プラットフォームでサポートされているインストール方法** 参照してください。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。 **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。

パラメーター	説明	値
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。	**false** または **true**

パラメーター	説明	値
	注記 Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal**、**External**、または **Mixed**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。API の公開ストラテジーと Ingress サーバーの公開ストラテジーが異なるクラスターをデプロイするには、**publish** を **Mixed** に設定 し、**operatorPublishingStrategy** パラメーターを使用します。

パラメーター	説明	値
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

重要

このパラメーターを **Manual** に設定すると、管理者レベルのシークレットを **kube-system** プロジェクトに保存する代替手段が有効になりますが、追加の設定手順が必要になります。詳細は、管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法を参照してください。

7.15.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

注記

デフォルトでは、**install-config.yaml** ファイルでアベイラビリティゾーンを指定すると、インストールプログラムはコントロールプレーンマシンとコンピューティングマシンを リージョン 内の これらのアベイラビリティゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。

表7.35 追加の Azure パラメーター

パラメーター	説明	値
compute: platform: azure: encryptionAtHost:	コンピュートマシンのホストレベルの暗号化を有効にします。ユーザー管理のサーバー側の暗号化とともに、この暗号化を有効にすることができます。この機能は、VM ホスト上の一時ディスク、エフェメラルディスク、キャッシュされたディスク、およびマネージド外のディスクを暗号化します。これは、ユーザー管理のサーバー側暗号化の前提条件ではありません。	**true** または **false**。デフォルトは **false** です。
compute: platform: azure: osDisk: diskSizeGB:	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
compute: platform: azure: osDisk: diskType:	ディスクのタイプを定義します。	**standard_LRS**、**premium_LRS**、または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
compute: platform: azure: ultraSSDCapability:	コンピュートノードの永続ストレージに Azure Ultra ディスクを使用できるようにします。これには、Azure リージョンおよびゾーンで Ultra ディスクを使用できるようにする必要があります。	**Enabled**、**Disabled**。デフォルトは **Disabled** です。
compute: platform: azure: osDisk: diskEncryptionSet: resourceGroup:	インストールの前提条件から設定されたディスク暗号化を含む Azure リソースグループの名前。このリソースグループは、クラスターが破棄されたときに Azure 暗号化キーが削除されないように、クラスターをインストールするリソースグループとは異なるものにする必要があります。この値は、ユーザー管理のディスク暗号化を使用してクラスターをインストールする場合にのみ必要です。	文字列 (例: **production_encryption_resource_group**)。

パラメーター	説明	値
compute: platform: azure: osDisk: diskEncryptionSet: name:	インストールの前提条件からの暗号化キーを含むディスク暗号化セットの名前。	文字列 (例: **production_disk_encryption_set**)。
compute: platform: azure: osDisk: diskEncryptionSet: subscriptionId:	ディスク暗号化セットが存在するディスク暗号化セットの Azure サブスクリプションを定義します。このセカンダリーディスク暗号化セットは、コンピュートマシンの暗号化に使用されます。	**00000000-0000-0000-0000-000000000000** 形式の文字列。
compute: platform: azure: osImage: publisher:	オプション: デフォルトで、インストールプログラムはコンピュートマシンの起動に使用する Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードしてインストールします。Azure Marketplace から入手できるカスタム RHCOS イメージを使用して、デフォルトの動作をオーバーライドできます。インストールプログラムは、このイメージをコンピュートマシンにのみ使用します。	文字列。イメージ発行者の名前。
compute: platform: azure: osImage: offer:	カスタム RHCOS イメージに関連付けられている Azure Marketplace オファーの名前。**compute.platform.azure.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。イメージオファーの名前。
compute: platform: azure: osImage: sku:	Azure Marketplace オファーのインスタンス。**compute.platform.azure.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。イメージオファーの SKU。

パラメーター	説明	値
compute: platform: azure: osImage: version:	イメージ SKU のバージョン番号。**compute.platform.azure.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。使用するイメージのバージョン。
compute: platform: azure: vmNetworkingType: 	Accelerated ネットワークを有効にします。Accelerated ネットワークにより、仮想マシンに対して Single Root I/O Virtualization (SR-IOV) が可能になり、ネットワークパフォーマンスが向上します。コンピュートマシンのインスタンスタイプが **Accelerated** ネットワークをサポートする場合、デフォルトでインストーラーは **Accelerated** ネットワークを有効にします。そうでない場合、デフォルトのネットワークタイプは **Basic** になります。	**Accelerated** または **Basic**。
compute: platform: azure: type:	コンピュートマシンの Azure インスタンスタイプを定義します。	String
compute: platform: azure: zones:	インストールプログラムがコンピュートマシンを作成するアベイラビリティーゾーン。	文字列リスト
compute: platform: azure: settings: securityType:	Confidential 仮想マシンまたはコンピュートノードの信頼された起動を有効にします。このオプションはデフォルトでは有効になっていません。	**ConfidentialVM** または **TrustedLaunch**。

パラメーター	説明	値
compute: platform: azure: settings: confidentialVM: uefiSettings: secureBoot:	Confidential 仮想マシンを使用している場合は、コンピュートノードでセキュアブートを有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
compute: platform: azure: settings: confidentialVM: uefiSettings: virtualizedTrustedPlatformModule:	Confidential 仮想マシンを使用している場合は、コンピュートノードで仮想化 Trusted Platform Module (vTPM) 機能を有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
compute: platform: azure: settings: trustedLaunch: uefiSettings: secureBoot:	信頼できる起動を使用している場合は、コンピュートノードでセキュアブートを有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
compute: platform: azure: settings: trustedLaunch: uefiSettings: virtualizedTrustedPlatformModule:	信頼された起動を使用している場合は、コンピュートノードで vTPM 機能を有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。

パラメーター	説明	値
compute: platform: azure: osDisk: securityProfile: securityEncryptionType:	コンピュートノードの仮想マシンゲスト状態の暗号化を有効にします。このパラメーターは、Confidential VM を使用する場合にのみ使用できます。	サポートされる値は **VMGuestStateOnly** のみです。
controlPlane: platform: azure: settings: securityType:	Confidential 仮想マシンまたはコントロールプレーンノードの信頼された起動を有効にします。このオプションはデフォルトでは有効になっていません。	**ConfidentialVM** または **TrustedLaunch**。
controlPlane: platform: azure: settings: confidentialVM: uefiSettings: secureBoot:	Confidential 仮想マシンを使用している場合は、コントロールプレーンノードでセキュアブートを有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
controlPlane: platform: azure: settings: confidentialVM: uefiSettings: virtualizedTrustedPlatformModule:	Confidential 仮想マシンを使用している場合は、コントロールプレーンノードで vTPM 機能を有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。

パラメーター	説明	値
controlPlane: platform: azure: settings: trustedLaunch: uefiSettings: secureBoot:	信頼できる起動を使用している場合は、コントロールプレーンノードでセキュアブートを有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
controlPlane: platform: azure: settings: trustedLaunch: uefiSettings: virtualizedTrustedPlatformModule:	信頼できる起動を使用している場合は、コントロールプレーンノードで vTPM 機能を有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
controlPlane: platform: azure: osDisk: securityProfile: securityEncryptionType:	コントロールプレーンノードの仮想マシンゲスト状態の暗号化を有効にします。このパラメーターは、Confidential VM を使用する場合にのみ使用できます。	サポートされる値は **VMGuestStateOnly** のみです。
controlPlane: platform: azure: type:	コントロールプレーンマシンの Azure インスタンスタイプを定義します。	String
controlPlane: platform: azure: zones:	インストールプログラムがコントロールプレーンマシンを作成するアベイラビリティーゾーン。	文字列リスト

パラメーター	説明	値
platform: azure: defaultMachinePlatform: settings: securityType:	すべてのノードに対して Confidential 仮想マシンまたは信頼された起動を有効にします。このオプションはデフォルトでは有効になっていません。	**ConfidentialVM** または **TrustedLaunch**。
platform: azure: defaultMachinePlatform: settings: confidentialVM: uefiSettings: secureBoot:	Confidential 仮想マシンを使用している場合は、すべてのノードでセキュアブートを有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
platform: azure: defaultMachinePlatform: settings: confidentialVM: uefiSettings: virtualizedTrustedPlatformModule:	Confidential 仮想マシンを使用している場合は、すべてのノードで仮想化 Trusted Platform Module (vTPM) 機能を有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。

パラメーター	説明	値
platform: azure: defaultMachinePlatform: settings: trustedLaunch: uefiSettings: secureBoot:	信頼できる起動を使用している場合は、すべてのノードでセキュアブートを有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
platform: azure: defaultMachinePlatform: settings: trustedLaunch: uefiSettings: virtualizedTrustedPlatformModule:	信頼された起動を使用している場合は、すべてのノードで vTPM 機能を有効にします。	**Enabled** または **Disabled**。デフォルトは **Disabled** です。
platform: azure: defaultMachinePlatform: osDisk: securityProfile: securityEncryptionType:	すべてのノードの仮想マシンのゲスト状態の暗号化を有効にします。このパラメーターは、Confidential VM を使用する場合にのみ使用できます。	サポートされる値は **VMGuestStateOnly** のみです。

パラメーター	説明	値
platform: azure: defaultMachinePlatform: encryptionAtHost:	コンピュートマシンのホストレベルの暗号化を有効にします。ユーザー管理のサーバー側の暗号化とともに、この暗号化を有効にすることができます。この機能は、VM ホスト上の一時ディスク、エフェメラルディスク、キャッシュされたディスク、および管理対象外のディスクを暗号化します。このパラメーターは、ユーザー管理のサーバー側暗号化の前提条件ではありません。	**true** または **false**。デフォルトは **false** です。
platform: azure: defaultMachinePlatform: osDisk: diskEncryptionSet: name:	インストールの前提条件からの暗号化キーを含むディスク暗号化セットの名前。	文字列 (例: **production_disk_encryption_set**)。
platform: azure: defaultMachinePlatform: osDisk: diskEncryptionSet: resourceGroup:	インストールの前提条件から設定されたディスク暗号化を含む Azure リソースグループの名前。このリソースグループは、クラスターが破棄されたときに Azure 暗号化キーが削除されないように、クラスターをインストールするリソースグループとは異なるものにする必要があります。この値は、ユーザー管理のディスク暗号化を使用してクラスターをインストールする場合にのみ必要です。	文字列 (例: **production_encryption_resource_group**)。
platform: azure: defaultMachinePlatform: osDisk: diskEncryptionSet: subscriptionId:	ディスク暗号化セットが存在するディスク暗号化セットの Azure サブスクリプションを定義します。このセカンダリーディスク暗号化セットは、コンピュートマシンの暗号化に使用されます。	**00000000-0000-0000-0000-000000000000** 形式の文字列。

パラメーター	説明	値
platform: azure: defaultMachinePlatform: osDisk: diskSizeGB:	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
platform: azure: defaultMachinePlatform: osDisk: diskType:	ディスクのタイプを定義します。	**premium_LRS** または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
platform: azure: defaultMachinePlatform: osImage: publisher:	オプション: デフォルトで、インストールプログラムは、コントロールプレーンおよびコンピュートマシンの起動に使用される Red Hat Enterprise Linux CoreOS (RHCOS)イメージをダウンロードしてインストールします。Azure Marketplace から入手できるカスタム RHCOS イメージを使用して、デフォルトの動作をオーバーライドできます。インストールプログラムは、両方のタイプのマシンにこのイメージを使用します。	文字列。イメージ発行者の名前。
platform: azure: defaultMachinePlatform: osImage: offer:	カスタム RHCOS イメージに関連付けられている Azure Marketplace オファーの名前。**platform.azure.defaultMachinePlatform.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。イメージオファーの名前。

パラメーター	説明	値
platform: azure: defaultMachinePlatform: osImage: sku:	Azure Marketplace オファーのインスタンス。**platform.azure.defaultMachinePlatform.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。イメージオファーの SKU。
platform: azure: defaultMachinePlatform: osImage: version:	イメージ SKU のバージョン番号。**platform.azure.defaultMachinePlatform.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。使用するイメージのバージョン。
platform: azure: defaultMachinePlatform: type:	コントロールプレーンおよびコンピュートマシンの Azure インスタンスタイプ。	Azure インスタンスタイプ。
platform: azure: defaultMachinePlatform: zones:	インストールプログラムがコンピュートマシンおよびコントロールプレーンマシンを作成するアベイラビリティーゾーン。	文字列リスト。
controlPlane: platform: azure: encryptionAtHost:	コントロールプレーンマシンのホストレベルの暗号化を有効にします。ユーザー管理のサーバー側の暗号化とともに、この暗号化を有効にすることができます。この機能は、VM ホスト上の一時ディスク、エフェメラルディスク、キャッシュされたディスク、およびマネージド外のディスクを暗号化します。これは、ユーザー管理のサーバー側暗号化の前提条件ではありません。	**true** または **false**。デフォルトは **false** です。

パラメーター	説明	値
controlPlane: platform: azure: osDisk: diskEncryptionSet: resourceGroup:	インストールの前提条件から設定されたディスク暗号化を含む Azure リソースグループの名前。このリソースグループは、クラスターが破棄されたときに Azure 暗号化キーが削除されないように、クラスターをインストールするリソースグループとは異なるものにする必要があります。この値は、ユーザー管理のディスク暗号化を使用してクラスターをインストールする場合にのみ必要です。	文字列 (例: **production_encryption_resource_group**)。
controlPlane: platform: azure: osDisk: diskEncryptionSet: name:	インストールの前提条件からの暗号化キーを含むディスク暗号化セットの名前。	文字列 (例: **production_disk_encryption_set**)。
controlPlane: platform: azure: osDisk: diskEncryptionSet: subscriptionId:	ディスク暗号化セットが存在するディスク暗号化セットの Azure サブスクリプションを定義します。このセカンダリーディスク暗号化セットは、コントロールプレーンマシンの暗号化に使用されます。	**00000000-0000-0000-0000-000000000000** 形式の文字列。
controlPlane: platform: azure: osDisk: diskSizeGB:	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **1024** です。
controlPlane: platform: azure: osDisk: diskType:	ディスクのタイプを定義します。	**premium_LRS** または **standardSSD_LRS**。デフォルトは **premium_LRS** です。

パラメーター	説明	値
controlPlane: platform: azure: osImage: publisher:	オプション: デフォルトで、インストールプログラムは、コントロールプレーンおよびコンピュートマシンの起動に使用する Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードしてインストールします。Azure Marketplace から入手できるカスタム RHCOS イメージを使用して、デフォルトの動作をオーバーライドできます。インストールプログラムは、このイメージをコントロールプレーンマシンにのみ使用します。	文字列。イメージ発行者の名前。
controlPlane: platform: azure: osImage: offer:	カスタム RHCOS イメージに関連付けられている Azure Marketplace オファーの名前。**controlPlane.platform.azure.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。イメージオファーの名前。
controlPlane: platform: azure: osImage: sku:	Azure Marketplace オファーのインスタンス。**controlPlane.platform.azure.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。イメージオファーの SKU。
controlPlane: platform: azure: osImage: version:	イメージ SKU のバージョン番号。**controlPlane.platform.azure.osImage.publisher** を使用する場合、このフィールドは必須です。	文字列。使用するイメージのバージョン。
controlPlane: platform: azure: ultraSSDCapability:	コントロールプレーンマシンの永続ストレージに Azure Ultra ディスクを使用できるようにします。これには、Azure リージョンおよびゾーンで Ultra ディスクを使用できるようにする必要があります。	**Enabled**、**Disabled**。デフォルトは **Disabled** です。

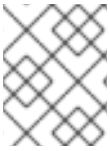
パラメーター	説明	値
controlPlane: platform: azure: vmNetworkingType: 	Accelerated ネットワークを有効にします。Accelerated ネットワークにより、仮想マシンに対して Single Root I/O Virtualization (SR-IOV) が可能になり、ネットワークパフォーマンスが向上します。コントロールプレーンマシンのインスタンスタイプが **Accelerated** ネットワークをサポートする場合、デフォルトでインストーラーは **Accelerated** ネットワークを有効にします。そうでない場合、デフォルトのネットワークタイプは **Basic** になります。	**Accelerated** または **Basic**。
platform: azure: baseDomainResourceGroupName: 	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform: azure: resourceGroupName: 	クラスターをインストールする既存のリソースグループの名前。このリソースグループは空で、この特定のクラスターにのみ使用する必要があります。クラスターコンポーネントは、リソースグループ内のすべてのリソースの所有権を想定します。インストールプログラムのサービスプリンシパルの範囲をこのリソースグループに制限する場合は、環境内でインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。	文字列 (例: **existing_resource_group**)。
platform: azure: outboundType: 	クラスターをインターネットに接続するために使用されるアウトバウンドルーティングストラテジー。ユーザー定義のルーティングを使用している場合、クラスターをインストールする前にアウトバウンドルーティングがすでに設定されている既存のネットワークが利用可能な状態にする必要があります。インストールプログラムはユーザー定義のルーティングの設定を行いません。**NatGateway** ルーティング戦略を指定した場合、インストールプログラムは NAT ゲートウェイを 1 つだ	**LoadBalancer**、**UserDefinedRouting**、または **NatGateway**。デフォルトは **LoadBalancer** です。

パラメーター	説明	値
	け作成します。**NatGateway** ルーティング戦略を指定する場合、アカウントには **Microsoft.Network/natGateways/read** パーミッションおよび **Microsoft.Network/natGateways/write** パーミッションが必要です。  **重要** **NatGateway** はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。 Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。	

パラメーター	説明	値
platform: azure: region:	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform: azure: zone:	マシンを配置するアベイラビリティーゾーンのリスト。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンのリスト (例: **["1", "2", "3"]**)。
platform: azure: customerManagedKey: keyVault: name:	Specifies the name of the key vault that contains the encryption key that is used to encrypt Azure storage.	文字列。
platform: azure: customerManagedKey: keyVault: keyName:	Azure ストレージの暗号化に使用するユーザー管理の暗号鍵の名前を指定します。	文字列。
platform: azure: customerManagedKey: keyVault: resourceGroup:	Key Vault とマネージド ID を含むリソースグループの名前を指定します。	文字列。

パラメーター	説明	値
platform: azure: customerManagedKey: keyVault: subscriptionId:	Key Vault に関連付けられたサブスクリプション ID を指定します。	**00000000-0000-0000-0000-000000000000** 形式の文字列。
platform: azure: customerManagedKey: userAssignedIdentityKey:	Key Vault を含むリソースグループに存在し、ユーザー管理の鍵へのアクセス権を持つユーザー割り当てのマネージド ID の名前を指定します。	文字列。
platform: azure: defaultMachinePlatform: ultraSSDCapability:	コントロールプレーンおよびコンピュートマシン上の永続ストレージに Azure Ultra ディスクを使用できるようにします。これには、Azure リージョンおよびゾーンで Ultra ディスクを使用できるようにする必要があります。	**Enabled**、**Disabled**。デフォルトは **Disabled** です。
platform: azure: networkResourceGroupName:	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。
platform: azure: virtualNetwork:	クラスターをデプロイする既存 VNet の名前。	文字列。
platform: azure: controlPlaneSubnet:	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。

パラメーター	説明	値
platform: azure: computeSubnet:	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform: azure: cloudName:	適切な Azure API エンドポイントで Azure SDK を設定するために使用される Azure クラウド環境の名前。空の場合、デフォルト値の **AzurePublicCloud** が使用されます。	**AzurePublicCloud** または **AzureUSGovernmentCloud** などの有効なクラウド環境。
platform: azure: defaultMachinePlatform: vmNetworkingType: 	Accelerated ネットワークを有効にします。Accelerated ネットワークにより、仮想マシンに対して Single Root I/O Virtualization (SR-IOV) が可能になり、ネットワークパフォーマンスが向上します。	**Accelerated** または **Basic**。コントロールプレーンマシンおよびコンピュートマシンのインスタンスタイプが **Accelerated** ネットワークをサポートする場合、デフォルトでインストーラーは **Accelerated** ネットワークを有効にします。そうでない場合、デフォルトのネットワークタイプは **Basic** になります。
operatorPublishingStrategy: apiserver:	API にサービスを提供するロードバランサーがパブリックかプライベートかを決定します。VNet の外部から API サーバーにアクセスできないようにするには、このパラメーターを **Internal** に設定します。VNet の外部から API サーバーにアクセスできるようにするには、このパラメーターを **External** に設定します。このパラメーターを設定する場合は、**publish** パラメーターを **Mixed** に設定する必要があります。	**External** または **Internal**。デフォルト値は **External** です。
operatorPublishingStrategy: ingress:	クラスターによって Ingress トラフィック用に作成される DNS リソースを公開するかどうかを決定します。Ingress 仮想 IP へのパブリックアクセスを防止するには、このパラメーターを **Internal** に設定します。Ingress 仮想 IP へのパブリックアクセスを可能にするには、このパラメーターを **External** に設定します。このパラメーターを設定する場合は、**publish** パラメーターを **Mixed** に設定する必要があります。	**External** または **Internal**。デフォルト値は **External** です。



注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

第8章 AZURE STACK HUB へのインストール

8.1. AZURE STACK HUB へのインストールの準備

8.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- Azure Stack Hub バージョン 2008 以降がインストールされている。

8.1.2. OpenShift Container Platform の Azure Stack Hub へのインストール要件

OpenShift Container Platform を Microsoft Azure にインストールする前に、Azure Stack Hub アカウントを設定する必要があります。

アカウントの設定、アカウントの制限、DNS ゾーン設定、必要なロール、およびサービスプリンシパルの作成の詳細は、Azure Stack Hub アカウントの設定 を参照してください。

8.1.3. Azure Stack Hub に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーによるプロビジョニングおよびユーザーによるプロビジョニングのインストールプロセスの詳細は、インストールプロセス を参照してください。

8.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

次の方法を使用して、OpenShift Container Platform インストールプログラムによってプロビジョニングされた Azure Stack Hub インフラストラクチャーにクラスターをインストールできます。

- **インストーラーでプロビジョニングされたインフラストラクチャーを使用した Azure Stack Hub へのクラスターのインストール**:OpenShift Container Platform インストールプログラムによってプロビジョニングされた Azure Stack Hub インフラストラクチャーに OpenShift Container Platform をインストールできます。

8.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法を使用して、独自にプロビジョニングする Azure Stack Hub インフラストラクチャーにクラスターをインストールできます。

- **ARM テンプレートを使用したクラスターの Azure Stack Hub へのインストール**: 独自に提供するインフラストラクチャーを使用して、OpenShift Container Platform を Azure Stack Hub にイ

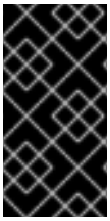
ンストールできます。提供される Azure Resource Manager (ARM) テンプレートを使用して、インストールを支援できます。

8.1.4. 次のステップ

- Azure Stack Hub アカウントの設定

8.2. AZURE STACK HUB アカウントの設定

OpenShift Container Platform をインストールする前に、Microsoft Azure アカウントを設定する必要があります。



重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

8.2.1. Azure Stack Hub アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure Stack Hub コンポーネントを使用し、デフォルトの Azure Stack Hub のクォータタイプ は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure Stack Hub コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	説明

コンポーネント	デフォルトで必要なコンポーネントの数	説明
vCPU	56	デフォルトのクラスターには 56 CPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップ、コントロールプレーン、およびワーカーマシンは 8 vCPU を使用する **Standard_DS4_v2** 仮想マシンを使用するため、デフォルトのクラスターには 56 vCPU が必要です。ブートストラップノードの仮想マシンはインストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。
VNet	1	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。
ネットワークインターフェイス	7	各デフォルトクラスターには、7 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。
ネットワークセキュリティーグループ	2	各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。

コンポーネント	デフォルトで必要なコンポーネントの数	説明
ネットワークロードバランサー	3	各クラスターは以下の ロードバランサー を作成します。 **default**: ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス **internal**: コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス **external**: コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス アプリケーションが追加の Kubernetes **LoadBalancer** サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。
パブリック IP アドレス	2	パブリックロードバランサーはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。
プライベート IP アドレス	7	内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。

関連情報

- ストレージの最適化

8.2.2. Azure Stack Hub での DNS ゾーンの設定

OpenShift Container Platform を Azure Stack Hub に正常にインストールするには、Azure Stack Hub DNS ゾーンに DNS レコードを作成する必要があります。DNS ゾーンはドメインに対する権威を持っている必要があります。レジストラーの DNS ゾーンを Azure Stack Hub に委譲するには、Microsoft の Azure Stack Hub データセンター DNS 統合 についてのドキュメントを参照してください。

8.2.3. 必要な Azure Stack Hub ロール

Microsoft Azure Stack Hub アカウントには、使用するサブスクリプションについて以下のロールが必要です。

- **Owner**

Azure ポータルでロールを設定するには、Microsoft ドキュメントの Manage access to resources in Azure Stack Hub with role-based access control を参照してください。

8.2.4. サービスプリンシパルの作成

OpenShift Container Platform とそのインストールプログラムは Azure Resource Manager を使用して Microsoft Azure リソースを作成するため、それを表すサービスプリンシパルを作成する必要があります。

前提条件

- Azure CLI のインストールまたは更新を実行します。
- Azure アカウントには、使用するサブスクリプションに必要なロールがなければなりません。

手順

1. 環境を登録します。

```
$ az cloud register -n AzureStackCloud --endpoint-resource-manager <endpoint> 1
```

1 Azure Resource Manager エンドポイント `https://management.<region>.<fqdn>/` を指定します。

詳細は、Microsoft のドキュメント を参照してください。

2. アクティブな環境を設定します。

```
$ az cloud set -n AzureStackCloud
```

3. Azure Stack Hub に特定の API バージョンを使用するように、環境設定を更新します。

```
$ az cloud update --profile 2019-03-01-hybrid
```

4. Azure CLI にログインします。

```
$ az login
```

マルチテナント環境の場合は、テナント ID も指定する必要があります。

5. Azure アカウントでサブスクリプションを使用している場合は、適切なサブスクリプションを使用していることを確認してください。

a. 利用可能なアカウントの一覧を表示し、クラスターに使用するサブスクリプションの **tenantId** の値を記録します。

```
$ az account list --refresh
```

出力例

```
[
  {
    "cloudName": AzureStackCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
```

```
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. アクティブなアカウントの詳細を表示し、**tenantId** 値が使用するサブスクリプションと一致することを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": AzureStackCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", 1
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

1 **tenantId** パラメーターの値が正しいサブスクリプション ID であることを確認してください。

c. 適切なサブスクリプションを使用していない場合には、アクティブなサブスクリプションを変更します。

```
$ az account set -s <subscription_id> 1
```

1 サブスクリプション ID を指定します。

d. サブスクリプション ID の更新を確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": AzureStackCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
```

```
    "type": "user"
  }
}
```

6. 出力から **tenantId** および **id** パラメーター値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

7. アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> \ 1
  --scopes /subscriptions/<subscription_id> 2
  --years <years> 3
```

1 サービスプリンシパル名を指定します。

2 サブスクリプション ID を指定します。

3 年数を指定します。デフォルトでは、サービスプリンシパルは 1 年で期限切れになります。**--years** オプションを使用すると、サービスプリンシパルの有効期間を延長できます。

出力例

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "displayName": <service_principal>",
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee"
}
```

8. 直前の出力の **appId** および **password** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

関連情報

- CCO モードの詳細は、Cloud Crednetial Operator について を参照してください。

8.2.5. 次のステップ

- OpenShift Container Platform クラスターをインストールします。
  - Azure Stack Hub にクラスターをすばやくインストールします。
  - ARM テンプレートを使用した Azure Stack Hub へのクラスターのインストール に従い、ユーザーによってプロビジョニングされるインフラストラクチャーでの Azure Stack Hub に OpenShift Container Platform クラスターをインストールします。

8.3. インストーラーでプロビジョニングされたインフラストラクチャーを使用して AZURE STACK HUB にクラスターをインストールします。

OpenShift Container Platform バージョン 4.16 では、installer-provisioned infrastructure を使用して、Microsoft Azure Stack Hub にクラスターをインストールできます。ただし、Azure Stack Hub に固有の値を指定するには、**install-config.yaml**ファイルを手動で設定する必要があります。

注記

インストールプログラムを使用して、インストーラーでプロビジョニングされたインフラストラクチャーを使用してクラスターをデプロイするときに、**azure** を選択できますが、このオプションは Azure Public Cloud でのみサポートされます。

8.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure Stack Hub アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- 約 16GB のローカルディスク容量があることを確認している。クラスターをインストールするには、RHCOS 仮想ハードディスク (VHD) クラスターイメージをダウンロードし、これを Azure Stack Hub 環境にアップロードして、デプロイメント中にアクセスできるようにする必要があります。VHD ファイルを解凍するには、これくらいのローカルディスク領域が必要です。

8.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

8.3.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

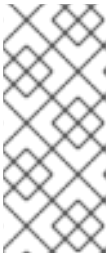
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

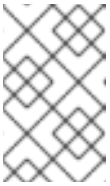
一部のディストリビューションでは、**~/.ssh/id_rsa** および **~/.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

8.3.4. RHCOS クラスターイメージのアップロード

RHCOS 仮想ハードディスク (VHD) クラスターイメージをダウンロードし、これを Azure Stack Hub 環境にアップロードして、デプロイメント中にアクセスできるようにする必要があります。

前提条件

- Azure アカウントを設定します。

手順

1. RHCOS VHD クラスターイメージを取得します。

   a. RHCOS VHD の URL を環境変数にエクスポートします。

   ```
   $ export COMPRESSED_VHD_URL=$(openshift-install coreos print-stream-json | jq -r
   '.architectures.x86_64.artifacts.azurestack.formats."vhd.gz".disk.location')
   ```

   b. 圧縮された RHCOS VHD ファイルをローカルにダウンロードします。

   ```
   $ curl -O -L ${COMPRESSED_VHD_URL}
   ```

2. VHD ファイルを展開します。

注記

展開した VHD ファイルは約 16 GB であるため、ホストシステムに 16 GB の空き領域があることを確認してください。VHD ファイルは、アップロードした後に削除できます。

3. ローカル VHD を Azure Stack Hub 環境にアップロードし、blob が公開されていることを確認します。たとえば、 **az** cli または Web ポータルを使用して VHD を blob にアップロードできます。

8.3.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

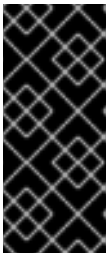
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。

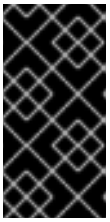
2. クラウドプロバイダーとして**Azure**を選択します。

3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

8.3.6. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

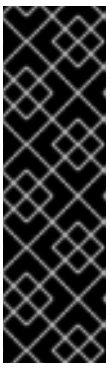
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

次の変更を行います。

a. 必要なインストールパラメーターを指定します。

b. **platform.azure** セクションを更新して、Azure Stack Hub に固有のパラメーターを指定します。

c. オプション:1 つ以上のデフォルト設定パラメーターを更新して、インストールをカスタマイズします。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- Azure Stack Hub のインストール設定パラメーター

8.3.6.1. Azure Stack Hub 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。これを使用して、手動で作成したインストール設定ファイルにパラメーター値を入力します。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Manual
controlPlane: (2) (3)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (4)
        diskType: premium_LRS
  replicas: 3
compute: (5)
- name: worker
  platform:
    azure:
      osDisk:
        diskSizeGB: 512 (6)
        diskType: premium_LRS
  replicas: 3
metadata:
  name: test-cluster (7) (8)
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 9
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    armEndpoint: azurestack_arm_endpoint 10 11
    baseDomainResourceGroupName: resource_group 12 13
    region: azure_stack_local_region 14 15
    resourceGroupName: existing_resource_group 16
    outboundType: Loadbalancer
    cloudName: AzureStackCloud 17
    clusterOSimage: https://vhdsa.blob.example.example.com/vhd/rhcos-410.84.202112040202-0-
azurestack.x86_64.vhd 18 19
pullSecret: '{"auths": ...}' 20 21
fips: false 22
sshKey: ssh-ed25519 AAAA... 23
additionalTrustBundle: | 24
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
```

1 7 10 12 14 17 18 20 必須。

2 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 6 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

8 クラスターの名前。

9 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

11 Azure Stack Hub オペレーターが提供する Azure Resource Manager エンドポイント。

13 ベースドメインの DNS ゾーンが含まれるリソースグループの名前。

15 Azure Stack Hub ローカルリージョンの名前。

16 クラスターをインストールする既存のリソースグループの名前。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

19 RHCOS VHD を含む Azure Stack 環境のストレージ blob の URL。

21 クラスターを認証するために必要なプルシークレット。

22 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

23 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

24 Azure Stack Hub 環境で内部認証局 (CA) を使用している場合は、CA 証明書を追加する必要があります。

8.3.7. クラウドクレデンシャルの手動管理

Cloud Credential Operator (CCO) は、手動モードのクラウドプロバイダーのみをサポートします。そのため、クラウドプロバイダーの ID およびアクセス管理 (IAM) シークレットを指定する必要があります。

手順

1. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

2. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
  ...
```

4. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
      ...
```

```
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```

重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

関連情報

- 手動で維持された認証情報によるクラウドプロバイダーリソースの更新

8.3.8. 内部 CA を使用するようにクラスターを設定する

Azure Stack Hub 環境で内部認証局 (CA) を使用している場合は、**cluster-proxy-01-config.yaml file** を更新して、内部 CA を使用するようにクラスターを設定します。

前提条件

- **install-config.yaml** ファイルを作成し、証明書の信頼バンドルを **.pem** 形式で指定します。
- クラスターマニフェストを作成します。

手順

1. インストールプログラムがファイルを作成するディレクトリーから、**manifests** ディレクトリーに移動します。
2. **user-ca-bundle** を **spec.trustedCA.name** フィールドに追加します。

   cluster-proxy-01-config.yaml ファイルの例

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  creationTimestamp: null
```

```
  name: cluster
spec:
  trustedCA:
    name: user-ca-bundle
status: {}
```

3. オプション: **manifests/ cluster-proxy-01-config.yaml** ファイルをバックアップします。クラスターをデプロイすると、インストールプログラムは **manifests/** ディレクトリーを消費します。

8.3.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

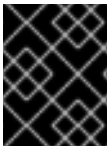
```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

8.3.10. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。

3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。

2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

8.3.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

8.3.12. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

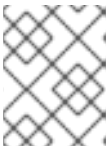
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

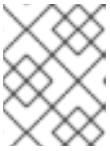


注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- Web コンソールへのアクセス

8.3.13. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデ

フォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

8.3.14. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

8.4. ネットワークをカスタマイズして AZURE STACK HUB にクラスターをインストールする

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが Azure Stack Hub 上にプロビジョニングするインフラストラクチャーに、カスタマイズしたネットワーク設定でクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

注記

インストールプログラムを使用して、インストーラーでプロビジョニングされたインフラストラクチャーを使用してクラスターをデプロイするときに、**azure** を選択できますが、このオプションは Azure Public Cloud でのみサポートされます。

8.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure Stack Hub アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- 約 16GB のローカルディスク容量があることを確認している。クラスターをインストールするには、RHCOS 仮想ハードディスク (VHD) クラスターイメージをダウンロードし、これを Azure Stack Hub 環境にアップロードして、デプロイメント中にアクセスできるようにする必要があります。VHD ファイルを解凍するには、これくらいのローカルディスク領域が必要です。

8.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

8.4.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

–

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

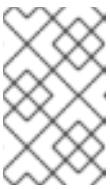
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

8.4.4. RHCOS クラスターイメージのアップロード

RHCOS 仮想ハードディスク (VHD) クラスターイメージをダウンロードし、これを Azure Stack Hub 環境にアップロードして、デプロイメント中にアクセスできるようにする必要があります。

前提条件

- Azure アカウントを設定します。

手順

1. RHCOS VHD クラスターイメージを取得します。
   a. RHCOS VHD の URL を環境変数にエクスポートします。

```
$ export COMPRESSED_VHD_URL=$(openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.azurestack.formats."vhd.gz".disk.location')
```

   b. 圧縮された RHCOS VHD ファイルをローカルにダウンロードします。

```
$ curl -O -L ${COMPRESSED_VHD_URL}
```

2. VHD ファイルを展開します。

注記

展開した VHD ファイルは約 16 GB であるため、ホストシステムに 16 GB の空き領域があることを確認してください。VHD ファイルは、アップロードした後に削除できます。

3. ローカル VHD を Azure Stack Hub 環境にアップロードし、blob が公開されていることを確認します。たとえば、 **az** cli または Web ポータルを使用して VHD を blob にアップロードできます。

8.4.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

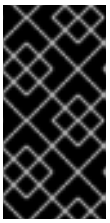
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. クラウドプロバイダーとして**Azure**を選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

8.4.6. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

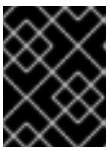
次の変更を行います。

a. 必要なインストールパラメーターを指定します。

b. **platform.azure** セクションを更新して、Azure Stack Hub に固有のパラメーターを指定します。

c. オプション: 1 つ以上のデフォルト設定パラメーターを更新して、インストールをカスタマイズします。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

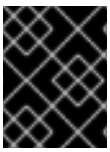
install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- Azure Stack Hub のインストール設定パラメーター

8.4.6.1. Azure Stack Hub 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。これを使用して、手動で作成したインストール設定ファイルにパラメーター値を入力します。

```
apiVersion: v1
baseDomain: example.com (1)
```

```
credentialsMode: Manual
controlPlane: (2) (3)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (4)
        diskType: premium_LRS
  replicas: 3
compute: (5)
- name: worker
  platform:
    azure:
      osDisk:
        diskSizeGB: 512 (6)
        diskType: premium_LRS
  replicas: 3
metadata:
  name: test-cluster (7) (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (9)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    armEndpoint: azurestack_arm_endpoint (10) (11)
    baseDomainResourceGroupName: resource_group (12) (13)
    region: azure_stack_local_region (14) (15)
    resourceGroupName: existing_resource_group (16)
    outboundType: Loadbalancer
    cloudName: AzureStackCloud (17)
    clusterOSimage: https://vhdsa.blob.example.example.com/vhd/rhcos-410.84.202112040202-0-
azurestack.x86_64.vhd (18) (19)
pullSecret: '{"auths": ...}' (20) (21)
fips: false (22)
sshKey: ssh-ed25519 AAAA... (23)
additionalTrustBundle: | (24)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
```

(1) (7) (10) (12) (14) (17) (18) (20) 必須。

(2) (5) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、**compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。

どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 6 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

8 クラスターの名前。

9 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

11 Azure Stack Hub オペレーターが提供する Azure Resource Manager エンドポイント。

13 ベースドメインの DNS ゾーンが含まれるリソースグループの名前。

15 Azure Stack Hub ローカルリージョンの名前。

16 クラスターをインストールする既存のリソースグループの名前。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

19 RHCOS VHD を含む Azure Stack 環境のストレージ blob の URL。

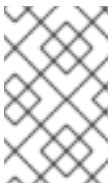
21 クラスターを認証するために必要なプルシークレット。

22 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

23 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

24 Azure Stack Hub 環境で内部認証局 (CA) を使用している場合は、CA 証明書を追加する必要があります。

8.4.7. クラウドクレデンシャルの手動管理

Cloud Credential Operator (CCO) は、手動モードのクラウドプロバイダーのみをサポートします。そのため、クラウドプロバイダーの ID およびアクセス管理 (IAM) シークレットを指定する必要があります。

手順

1. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

2. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
  ...
```

4. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて

spec.secretRef に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```

重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

関連情報

- Web コンソールを使用してクラスターを更新
- CLI を使用したクラスターの更新

8.4.8. 内部 CA を使用するようにクラスターを設定する

Azure Stack Hub 環境で内部認証局 (CA) を使用している場合は、**cluster-proxy-01-config.yaml file** を更新して、内部 CA を使用するようにクラスターを設定します。

前提条件

- **install-config.yaml** ファイルを作成し、証明書の信頼バンドルを **.pem** 形式で指定します。
- クラスターマニフェストを作成します。

手順

1. インストールプログラムがファイルを作成するディレクトリーから、**manifests** ディレクトリーに移動します。
2. **user-ca-bundle** を **spec.trustedCA.name** フィールドに追加します。

   cluster-proxy-01-config.yaml ファイルの例

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  creationTimestamp: null
  name: cluster
spec:
  trustedCA:
    name: user-ca-bundle
status: {}
```

3. オプション: **manifests/ cluster-proxy-01-config.yaml** ファイルをバックアップします。クラスターをデプロイすると、インストールプログラムは **manifests/** ディレクトリーを消費します。

8.4.9. ネットワーク設定フェーズ

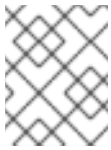
OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**

これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

重要

CIDR 範囲 **172.17.0.0/16** は libVirt によって予約されています。この範囲、またはこの範囲と重複する範囲をクラスター内のネットワークに使用することはできません。

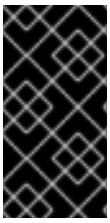
フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 でネットワークプラグインをさらにカスタマイズできます。

8.4.10. 高度なネットワーク設定の指定

ネットワークプラグインに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。



重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 次の例のように、**cluster-network-03-config.yml** ファイルでクラスターの高度なネットワーク設定を指定します。

OVN-Kubernetes ネットワークプロバイダーを有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

8.4.11. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

8.4.11.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表8.1 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

フィールド	型	説明
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定

defaultNetwork オブジェクトの値は、以下の表で定義されます。

表8.2 defaultNetwork オブジェクト

フィールド	型	説明

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表8.3 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。

フィールド	型	説明
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表8.4 ovnKubernetesConfig.ipv4 object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。

フィールド	型	説明
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表8.5 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表8.6 **policyAuditConfig オブジェクト**

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。

フィールド	型	説明
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表8.7 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。

フィールド	型	説明
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表8.8 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表8.9 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表8.10 **ipsecConfig** オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
```

```
type: OVNKubernetes
ovnKubernetesConfig:
  mtu: 1400
  genevePort: 6081
    ipsecConfig:
      mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表8.11 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: `kubeProxyConfig:` `  proxyArguments:` `  iptables-min-sync-period:` `  - 0s`

8.4.12. OVN-Kubernetes を使用したハイブリッドネットワークの設定

OVN-Kubernetes ネットワークプラグインを使用してハイブリッドネットワークを使用するようにクラスターを設定できます。これにより、異なるノードのネットワーク設定をサポートするハイブリッドクラスターが可能になります。

前提条件

- **install-config.yaml** ファイルで **networking.networkType** パラメーターの **OVNKubernetes** を

定義していること。詳細は、選択したクラウドプロバイダーでの OpenShift Container Platform ネットワークのカスタマイズの設定についてのインストールドキュメントを参照してください。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

ここでは、以下のようになります。

<installation_directory>
クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
EOF
```

ここでは、以下のようになります。

<installation_directory>
クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

3. **cluster-network-03-config.yml** ファイルをエディターで開き、以下の例のようにハイブリッドネットワークで OVN-Kubernetes を設定します。

ハイブリッドネットワーク設定の指定

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork: 1
        - cidr: 10.132.0.0/14
          hostPrefix: 23
```

1 追加のオーバーレイネットワーク上のノードに使用される CIDR 設定を指定します。**hybridClusterNetwork** CIDR は **clusterNetwork** CIDR と重複できません。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

8.4.13. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

8.4.14. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

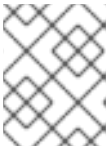
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

8.4.15. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

8.4.16. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートをリスト表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- Accessing the web console.

8.4.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 し

て、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

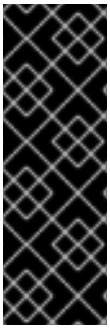
8.4.18. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

8.5. ARM テンプレートを使用したクラスターの AZURE STACK HUB へのインストール

OpenShift Container Platform バージョン 4.16 では、独自に提供するインフラストラクチャーを使用して、Microsoft Azure Stack Hub にクラスターをインストールできます。

これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Azure Resource Manager (ARM) テンプレートが提供されます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の ARM テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

8.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように Azure Stack Hub アカウントを設定 している。
- Azure CLI をダウンロードし、これをコンピューターにインストールしている。Azure ドキュメントの Install the Azure CLI を参照してください。以下のドキュメントについては、Azure CLI のバージョン **2.28.0** を使用してテストされていますAzure CLI コマンドは、使用するバージョンによって動作が異なる場合があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

8.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

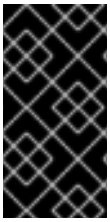
- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

8.5.3. Azure Stack Hub プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするために Azure プロジェクトを設定する必要があります。



重要

パブリックエンドポイントで利用可能なすべての Azure Stack Hub リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure Stack Hub が制限する語のリストは、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

8.5.3.1. Azure Stack Hub アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure Stack Hub コンポーネントを使用し、デフォルトの Azure Stack Hub のクォータタイプ は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure Stack Hub コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	説明
vCPU	56	デフォルトのクラスターには 56 CPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップ、コントロールプレーン、およびワーカーマシンは 8 vCPU を使用する **Standard_DS4_v2** 仮想マシンを使用するため、デフォルトのクラスターには 56 vCPU が必要です。ブートストラップノードの仮想マシンはインストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。
VNet	1	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。
ネットワークインターフェイス	7	各デフォルトクラスターには、7 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。
ネットワークセキュリティーグループ	2	各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。

<table>
<tr><th>コンポーネント</th><th>デフォルトで必要なコンポーネントの数</th><th>説明</th></tr>
<tr><td>ネットワークロードバランサー</td><td>3</td><td>各クラスターは以下の ロードバランサー を作成します。<table><tr><td>default</td><td>ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス</td></tr><tr><td>internal</td><td>コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス</td></tr><tr><td>external</td><td>コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス</td></tr></table>アプリケーションが追加の Kubernetes LoadBalancer サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。</td></tr>
<tr><td>パブリック IP アドレス</td><td>2</td><td>パブリックロードバランサーはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。</td></tr>
<tr><td>プライベート IP アドレス</td><td>7</td><td>内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。</td></tr>
</table>

関連情報

- ストレージの最適化

8.5.3.2. Azure Stack Hub での DNS ゾーンの設定

OpenShift Container Platform を Azure Stack Hub に正常にインストールするには、Azure Stack Hub DNS ゾーンに DNS レコードを作成する必要があります。DNS ゾーンはドメインに対する権威を持っている必要があります。レジストラーの DNS ゾーンを Azure Stack Hub に委譲するには、Microsoft の Azure Stack Hub データセンター DNS 統合 についてのドキュメントを参照してください。

この DNS ゾーンの作成例 を参照し、Azure の DNS ソリューションを確認することができます。

8.5.3.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認しま

す。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

8.5.3.4. 必要な Azure Stack Hub ロール

Microsoft Azure Stack Hub アカウントには、使用するサブスクリプションについて以下のロールが必要です。

- **Owner**

Azure ポータルでロールを設定するには、Microsoft ドキュメントの Manage access to resources in Azure Stack Hub with role-based access control を参照してください。

8.5.3.5. サービスプリンシパルの作成

OpenShift Container Platform とそのインストールプログラムは Azure Resource Manager を使用して Microsoft Azure リソースを作成するため、それを表すサービスプリンシパルを作成する必要があります。

前提条件

- Azure CLI のインストールまたは更新を実行します。
- Azure アカウントには、使用するサブスクリプションに必要なロールがなければなりません。

手順

1. 環境を登録します。

```
$ az cloud register -n AzureStackCloud --endpoint-resource-manager <endpoint> 1
```

1 Azure Resource Manager エンドポイント `https://management.<region>.<fqdn>/` を指定します。

詳細は、Microsoft のドキュメント を参照してください。

2. アクティブな環境を設定します。

```
$ az cloud set -n AzureStackCloud
```

3. Azure Stack Hub に特定の API バージョンを使用するように、環境設定を更新します。

```
$ az cloud update --profile 2019-03-01-hybrid
```

4. Azure CLI にログインします。

```
$ az login
```

マルチテナント環境の場合は、テナント ID も指定する必要があります。

5. Azure アカウントでサブスクリプションを使用している場合は、適切なサブスクリプションを使用していることを確認してください。

a. 利用可能なアカウントの一覧を表示し、クラスターに使用するサブスクリプションの **tenantId** の値を記録します。

```
$ az account list --refresh
```

出力例

```
[
  {
    "cloudName": AzureStackCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. アクティブなアカウントの詳細を表示し、**tenantId** 値が使用するサブスクリプションと一致することを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": AzureStackCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** パラメーターの値が正しいサブスクリプション ID であることを確認してください。

c. 適切なサブスクリプションを使用していない場合には、アクティブなサブスクリプションを変更します。

```
$ az account set -s <subscription_id> (1)
```

(1) サブスクリプション ID を指定します。

d. サブスクリプション ID の更新を確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": AzureStackCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

6. 出力から **tenantId** および **id** パラメーター値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

7. アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> \ 1
  --scopes /subscriptions/<subscription_id> 2
  --years <years> 3
```

1 サービスプリンシパル名を指定します。

2 サブスクリプション ID を指定します。

3 年数を指定します。デフォルトでは、サービスプリンシパルは 1 年で期限切れになります。**--years** オプションを使用すると、サービスプリンシパルの有効期間を延長できます。

出力例

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "displayName": <service_principal>",
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee"
}
```

8. 直前の出力の **appId** および **password** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

関連情報

- CCO モードの詳細は、Cloud Crednetial Operator について を参照してください。

8.5.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

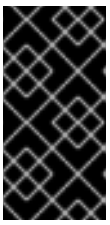
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. クラウドプロバイダーとして**Azure**を選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

8.5.5. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各

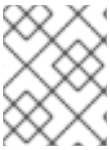
ノードの **core** ユーザーの ~/**.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

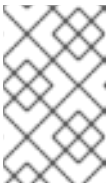
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

8.5.6. Azure Stack Hub のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Microsoft Azure Stack Hub にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイルを手動で作成し、Kubernetes マニフェストおよび Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

8.5.6.1. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

Azure Stack Hub について以下の変更を加えます。

a. **compute** プールの **replicas** パラメーターを **0** に設定します。

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 1
```

1 **0** に設定します。

コンピュートマシンは後で手動でプロビジョニングされます。

b. **install-config.yaml** ファイルの **platform.azure** セクションを更新し、Azure Stack Hub 設定を設定します。

```
platform:
  azure:
    armEndpoint: <azurestack_arm_endpoint> 1
```

```
baseDomainResourceGroupName: <resource_group> 2
cloudName: AzureStackCloud 3
region: <azurestack_region> 4
```

1. **https://adminmanagement.local.azurestack.external** など、Azure Stack Hub 環境の Azure Resource Manager エンドポイントを指定します。
2. ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。
3. 適切な Azure API エンドポイントで Azure SDK を設定するために使用される Azure Stack Hub 環境を指定します。
4. Azure Stack Hub リージョンの名前を指定します。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

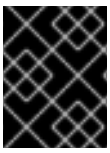
install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- Azure Stack Hub のインストール設定パラメーター

8.5.6.2. Azure Stack Hub 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。これを使用して、手動で作成したインストール設定ファイルにパラメーター値を入力します。

```
apiVersion: v1
baseDomain: example.com
controlPlane: 1
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 2
        diskType: premium_LRS
  replicas: 3
compute: 3
- name: worker
  platform:
    azure:
      osDisk:
        diskSizeGB: 512 4
```

```
        diskType: premium_LRS
  replicas: 0
metadata:
  name: test-cluster 5
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 6
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    armEndpoint: azurestack_arm_endpoint 7
    baseDomainResourceGroupName: resource_group 8
    region: azure_stack_local_region 9
    resourceGroupName: existing_resource_group 10
    outboundType: Loadbalancer
    cloudName: AzureStackCloud 11
pullSecret: '{"auths": ...}' 12
fips: false 13
additionalTrustBundle: | 14
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
sshKey: ssh-ed25519 AAAA... 15
```

1 3 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

2 4 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノードの最小推奨値は 1024 GB です。

5 クラスターの名前を指定します。

6 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

7 Azure Stack Hub Operator が提供する Azure Resource Manager エンドポイントを指定します。

8 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

9 Azure Stack Hub ローカルリージョンの名前を指定します。

10 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

11 Azure Stack Hub 環境をターゲットプラットフォームとして指定します。

12 クラスターの認証に必要なプルシークレットを指定します。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat

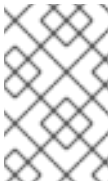
重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

14 Azure Stack Hub 環境で内部認証局 (CA) を使用している場合は、必要な証明書バンドルを **.pem** 形式で追加します。

15 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

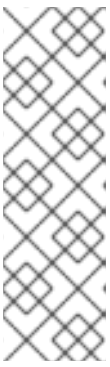
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

8.5.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

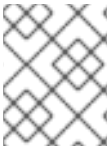
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

8.5.6.4. ARM テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Microsoft Azure Stack Hub で実行するのに役立つ指定の Azure Resource Manager (ARM) テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。



注記

特定の ARM テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. 提供される ARM テンプレートで使用される **install-config.yaml** にある一般的な変数をエクスポートします。

```
$ export CLUSTER_NAME=<cluster_name>(1)
$ export AZURE_REGION=<azure_region>(2)
$ export SSH_KEY=<ssh_key>(3)
$ export BASE_DOMAIN=<base_domain>(4)
$ export BASE_DOMAIN_RESOURCE_GROUP=<base_domain_resource_group>(5)
```

(1) **install-config.yaml** ファイルからの **.metadata.name** 属性の値。

(2) クラスターをデプロイするリージョンを選択します。これは、**install-config.yaml** ファイルからの **.platform.azure.region** 属性の値です。

(3) 文字列としての SSH RSA 公開鍵ファイル。SSH キーは、スペースが含まれているために引用符で囲む必要があります。これは、**install-config.yaml** ファイルからの **.sshKey** 属性の値です。

(4) クラスターをデプロイするベースドメイン。ベースドメインは、クラスターに作成した DNS ゾーンに対応します。これは、**install-config.yaml** からの **.baseDomain** 属性の値です。

(5) DNS ゾーンが存在するリソースグループ。これは、**install-config.yaml** ファイルからの **.platform.azure.baseDomainResourceGroupName** 属性の値です。

以下に例を示します。

```
$ export CLUSTER_NAME=test-cluster
$ export AZURE_REGION=centralus
$ export SSH_KEY="ssh-rsa xxx/xxx/xxx= user@email.com"
$ export BASE_DOMAIN=example.com
$ export BASE_DOMAIN_RESOURCE_GROUP=ocp-cluster
```

2. kubeadmin 認証情報をエクスポートします。

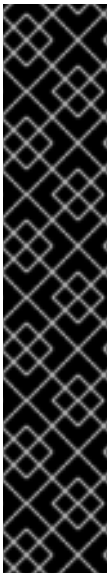
```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

8.5.6.5. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

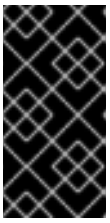
これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

4. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```



重要

ユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールするときに **MachineAPI** 機能を無効にした場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除する必要があります。そうしないと、クラスターのインストールに失敗します。

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
```

```
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. オプション: Azure Stack Hub 環境で内部認証局 (CA) を使用する場合には、**<installation_directory>/manifests/cluster-proxy-01-config.yaml** の **.spec.trustedCA.name** フィールドを更新して、**user-ca-bundle** を使用する必要があります。

```
...
spec:
  trustedCA:
    name: user-ca-bundle
...
```

後で、CA を含めるようにブートストラップ Ignition を更新する必要があります。

8. ユーザーによってプロビジョニングされるインフラストラクチャーで Azure を設定する場合、Azure Resource Manager (ARM) テンプレートで後に使用するためにマニフェストファイルに定義された一般的な変数の一部をエクスポートする必要があります。

   a. 以下のコマンドを使用してインフラストラクチャー ID をエクスポートします。

```
$ export INFRA_ID=<infra_id> 1
```

   1 OpenShift Container Platform クラスターには、**<cluster_name>-<random_string>** の形式の識別子 (**INFRA_ID**) が割り当てられます。これは、提供される ARM テンプレートを使用して作成されるほとんどのリソースのベース名として使用されます。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.infrastructureName** 属性の値です。

   b. 以下のコマンドを使用してリソースグループをエクスポートします。

```
$ export RESOURCE_GROUP=<resource_group> 1
```

   1 この Azure デプロイメントで作成されたすべてのリソースは、リソースグループ の一部として存在します。リソースグループ名は、**<cluster_name>-<random_string>-rg** 形式の **INFRA_ID** をベースとしています。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.platformStatus.azure.resourceGroupName** 属性の値です。

9. クラウド認証情報を手動で作成します。

   a. インストールプログラムが含まれるディレクトリーから、**openshift-install** バイナリーがビルドされる OpenShift Container Platform リリースイメージの詳細を取得します。

```
$ openshift-install version
```

出力例

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

b. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

c. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

2 **install-config.yaml** ファイルの場所を指定します。

3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-azure
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
```

d. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクト

について **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。シークレットデータの形式は、クラウドプロバイダーごとに異なります。

secrets.yaml ファイルのサンプル：

```
apiVersion: v1
kind: Secret
metadata:
  name: ${secret_name}
  namespace: ${secret_namespace}
stringData:
  azure_subscription_id: ${subscription_id}
  azure_client_id: ${app_id}
  azure_client_secret: ${client_secret}
  azure_tenant_id: ${tenant_id}
  azure_resource_prefix: ${cluster_name}
  azure_resourcegroup: ${resource_group}
  azure_region: ${azure_region}
```

e. Cloud Credential Operator (CCO) を無効にして manifests ディレクトリーに **cco-configmap.yaml** ファイルを作成します。

サンプル ConfigMap オブジェクト

```
apiVersion: v1
kind: ConfigMap
metadata:
name: cloud-credential-operator-config
namespace: openshift-cloud-credential-operator
  annotations:
    release.openshift.io/create-only: "true"
data:
  disabled: "true"
```

10. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- クラウドクレデンシャルの手動管理

8.5.6.6. オプション: 別個の /var パーティションの作成

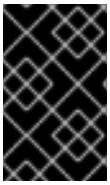
OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。



重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

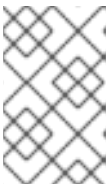
4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の /**var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

8.5.7. Azure リソースグループの作成

Microsoft Azure リソースグループ を作成する必要があります。これは Azure Stack Hub での OpenShift Container Platform クラスターのインストール時に使用されます。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

- サポートされる Azure リージョンにリソースグループを作成します。

```
$ az group create --name ${RESOURCE_GROUP} --location ${AZURE_REGION}
```

8.5.8. RHCOS クラスターイメージおよびブートストラップ Ignition 設定ファイルのアップロード

Azure クライアントは、ローカルに存在するファイルに基づくデプロイメントをサポートしていません。RHCOS 仮想ハードディスク (VHD) クラスターイメージとブートストラップ Ignition 設定ファイルをコピーしてストレージコンテナーに保存し、デプロイメント中にアクセスできるようにする必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. VHD クラスターイメージを保存するために Azure ストレージアカウントを作成します。

```
$ az storage account create -g ${RESOURCE_GROUP} --location ${AZURE_REGION} --
name ${CLUSTER_NAME}sa --kind Storage --sku Standard_LRS
```

警告

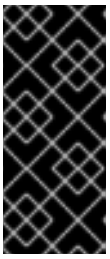
Azure ストレージアカウント名は 3 文字から 24 文字の長さで、数字および小文字のみを使用する必要があります。**CLUSTER_NAME** 変数がこれらの制限に準拠しない場合、Azure ストレージアカウント名を手動で定義する必要があります。Azure ストレージアカウント名の制限についての詳細は、Azure ドキュメントの Resolve errors for storage account names を参照してください。

2. ストレージアカウントキーを環境変数としてエクスポートします。

```
$ export ACCOUNT_KEY=`az storage account keys list -g ${RESOURCE_GROUP} --
account-name ${CLUSTER_NAME}sa --query "[0].value" -o tsv`
```

3. RHCOS VHD の URL を環境変数にエクスポートします。

```
$ export COMPRESSED_VHD_URL=$(openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.azurestack.formats."vhd.gz".disk.location')
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージを指定する必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

4. VHD のストレージコンテナーを作成します。

```
$ az storage container create --name vhd --account-name ${CLUSTER_NAME}sa --account-
key ${ACCOUNT_KEY}
```

5. 圧縮された RHCOS VHD ファイルをローカルにダウンロードします。

```
$ curl -O -L ${COMPRESSED_VHD_URL}
```

6. VHD ファイルを展開します。

注記

展開した VHD ファイルは約 16 GB であるため、ホストシステムに 16 GB の空き領域があることを確認してください。VHD ファイルはアップロード後に削除できます。

7. ローカル VHD を blob にコピーします。

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key
${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -f rhcos-<rhcos_version>-azurestack.x86_64.vhd
```

8. blob ストレージコンテナーを作成し、生成された **bootstrap.ign** ファイルをアップロードします。

```
$ az storage container create --name files --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY}
```

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key
${ACCOUNT_KEY} -c "files" -f "<installation_directory>/bootstrap.ign" -n "bootstrap.ign"
```

8.5.9. DNS ゾーンの作成例

DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターに必要です。シナリオに適した DNS ストラテジーを選択する必要があります。

この例では、Azure Stack Hub のデータセンター DNS 統合 が使用されるため、DNS ゾーンが作成されます。

注記

DNS ゾーンは、クラスターデプロイメントと同じリソースグループに存在している必要はなく、必要なベースドメイン用にすでに組織内に存在している可能性があります。その場合、DNS ゾーンの作成を省略できます。先に生成したインストール設定がこのシナリオに基づいていることを確認してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

- **BASE_DOMAIN_RESOURCE_GROUP** 環境変数でエクスポートされたリソースグループに、新規 DNS ゾーンを作成します。

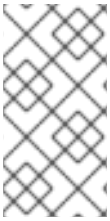
```
$ az network dns zone create -g ${BASE_DOMAIN_RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

すでに存在する DNS ゾーンを使用している場合は、この手順を省略できます。

での DNS ゾーンの設定 についてのセクションを参照してください。

8.5.10. Azure Stack Hub での VNet の作成

OpenShift Container Platform クラスター用に Microsoft Azure Stack Hub で使用する仮想ネットワーク (VNet) を作成する必要があります。各種の要件を満たすように VPC をカスタマイズできます。VNet を作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

提供される ARM テンプレートを使用して Azure Stack Hub インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VNet の ARM テンプレート**セクションからテンプレートをコピーし、これを **01_vnet.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要な VNet について記述しています。
2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/01_vnet.json" \
  --parameters baseName="${INFRA_ID}" 1
```

1 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

8.5.10.1. VNet の ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な VNet をデプロイすることができます。

例8.101_vnet.json ARM テンプレート

link:https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/azurestack/01_vnet.json[]

8.5.11. Azure Stack Hub インフラストラクチャー用の RHCOS クラスターイメージのデプロイ

OpenShift Container Platform ノードに Microsoft Azure Stack Hub 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- RHCOS 仮想ハードディスク (VHD) クラスターイメージを Azure ストレージコンテナーに保存します。
- ブートストラップ Ignition 設定ファイルを Azure ストレージコンテナーに保存します。

手順

1. 本トピックの **イメージストレージの ARM テンプレート**セクションからテンプレートをコピーし、これを **02_storage.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なイメージストレージ について記述しています。
2. RHCOS VHD blob URL を変数としてエクスポートします。

```
$ export VHD_BLOB_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -o tsv`
```

3. クラスターイメージのデプロイ

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/02_storage.json" \
  --parameters vhdBlobURL="${VHD_BLOB_URL}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameters storageAccount="${CLUSTER_NAME}sa" \ 3
  --parameters architecture="<architecture>" 4
```

1. マスターマシンおよびワーカーマシンを作成するために使用される RHCOS VHD の blob URL。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. Azure ストレージアカウントの名前。
4. システムアーキテクチャーを指定します。有効な値は、**x64** (デフォルト) または **Arm64** です。

8.5.11.1. イメージストレージの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な保存された Red Hat Enterprise Linux CoreOS (RHCOS) をデプロイすることができます。

例8.2 02_storage.json ARM テンプレート

```
link:https://raw.githubusercontent.com/openshift/installer/release-
4.16/upi/azurestack/02_storage.json[]
```

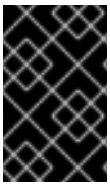
8.5.12. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

8.5.12.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表8.12 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット

プロトコル	ポート	説明
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表8.13 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表8.14 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

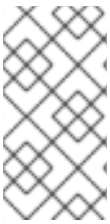
プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

8.5.13. Azure Stack Hub でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用するネットワークおよび負荷分散を Microsoft Azure Stack Hub で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

負荷分散には、以下の DNS レコードが必要です。

- DNS ゾーンの API パブリックロードバランサーの **api** DNS レコード
- DNS ゾーンの API 内部ロードバランサーの **api-int** DNS レコード



注記

提供される ARM テンプレートを使用して Azure Stack Hub インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure Stack Hub で VNet および関連するサブネットを作成し、設定します。

手順

1. 本トピックの **ネットワークおよびロードバランサーの ARM テンプレート**セクションからテンプレートをコピーし、これを **03_infra.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。

2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/03_infra.json" \
  --parameters baseName="${INFRA_ID}" 1
```

1 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. **api** DNS レコードおよび **api-int** DNS レコードを作成します。API DNS レコードの作成時に、**${BASE_DOMAIN_RESOURCE_GROUP}** 変数は DNS ゾーンが存在するリソースグループを参照する必要があります。

   a. 以下の変数をエクスポートします。

```
$ export PUBLIC_IP=`az network public-ip list -g ${RESOURCE_GROUP} --query "[?
name=='${INFRA_ID}-master-pip'] | [0].ipAddress" -o tsv`
```

   b. 以下の変数をエクスポートします。

```
$ export PRIVATE_IP=`az network lb frontend-ip show -g "$RESOURCE_GROUP" --lb-
name "${INFRA_ID}-internal" -n internal-lb-ip --query "privateIpAddress" -o tsv`
```

   c. 新しい DNS ゾーンに **api** DNS レコードを作成します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n api -a ${PUBLIC_IP} --ttl 60
```

   クラスターを既存の DNS ゾーンに追加する場合は、**api** DNS レコードを代わりに作成できます。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n api.${CLUSTER_NAME} -a ${PUBLIC_IP} --ttl 60
```

   d. 新しい DNS ゾーンに **api-int** DNS レコードを作成します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z "${CLUSTER_NAME}.${BASE_DOMAIN}" -n api-int -a ${PRIVATE_IP} --ttl 60
```

   クラスターを既存の DNS ゾーンに追加する場合は、**api-int** DNS レコードを代わりに作成できます。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n api-int.${CLUSTER_NAME} -a ${PRIVATE_IP} --ttl 60
```

8.5.13.1. ネットワークおよびロードバランサーの ARM テンプレート

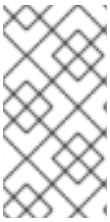
以下の Azure Resource Manager (ARM) テンプレートを使用して、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例8.3 03_infra.json ARM テンプレート

link:https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/azurestack/03_infra.json[]

8.5.14. Azure Stack Hub でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Microsoft Azure Stack Hub で作成する必要があります。このマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

提供されている ARM テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure Stack Hub で VNet および関連するサブネットを作成し、設定します。
- Azure Stack Hub でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。

手順

1. 本トピックの **ブートストラップマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. ブートストラップ URL 変数をエクスポートします。

```
$ bootstrap_url_expiry=`date -u -d "10 hours" '+%Y-%m-%dT%H:%MZ'`
```

```
$ export BOOTSTRAP_URL=`az storage blob generate-sas -c 'files' -n 'bootstrap.ign' --https-only --full-uri --permissions r --expiry $bootstrap_url_expiry --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -o tsv`
```

3. ブートストラップ Ignition 変数をエクスポートします。
   a. ご使用の環境で公開認証局 (CA) を使用している場合は、次のコマンドを実行します。

```
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "3.2.0" --arg url
${BOOTSTRAP_URL} '{ignition:{version:$v,config:{replace:{source:$url}}}}' | base64 | tr -
d '\n'`
```

b. 環境で内部 CA を使用している場合は、PEM エンコードバンドルをブートストラップ Ignition スタブに追加して、ブートストラップ仮想マシンがストレージアカウントからブートストラップ Ignition をプルできるようにする必要があります。次のコマンドを実行します。これは、CA が **CA.pem** のファイルにあることを前提としています。

```
$ export CA="data:text/plain;charset=utf-8;base64,$(cat CA.pem |base64 |tr -d '\n')"
```

```
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "3.2.0" --arg url
"$BOOTSTRAP_URL" --arg cert "$CA" '{ignition:{version:$v,security:{tls:
{certificateAuthorities:[{source:$cert}]}},config:{replace:{source:$url}}}}' | base64 | tr -d
'\n'`
```

4. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create --verbose -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/04_bootstrap.json" \
  --parameters bootstrapIgnition="${BOOTSTRAP_IGNITION}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameters diagnosticsStorageAccountName="${CLUSTER_NAME}sa" 3
```

1. ブートストラップクラスターのブートストラップ Ignition コンテンツ。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. クラスターのストレージアカウントの名前。

8.5.14.1. ブートストラップマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例8.4 04_bootstrap.json ARM テンプレート

link:https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/azurestack/04_bootstrap.json[]

8.5.15. Azure Stack Hub でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Microsoft Azure Stack Hub で作成する必要があります。これらのマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

提供される ARM テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合は、インストールログで Red Hat サポートに接続することを検討してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure Stack Hub で VNet および関連するサブネットを作成し、設定します。
- Azure Stack Hub でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの **コントロールプレーンマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **05_masters.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. コントロールプレーンマシンのデプロイメントに必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/05_masters.json" \
  --parameters masterIgnition="${MASTER_IGNITION}" \ 1
  --parameters baseName="${INFRA_ID}" \ 2
  --parameters diagnosticsStorageAccountName="${CLUSTER_NAME}sa" 3
```

1. コントロールプレーンノード (別名マスターノード) の Ignition コンテンツ。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。
3. クラスターのストレージアカウントの名前。

8.5.15.1. コントロールプレーンマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例8.5 05_masters.json ARM テンプレート

link:https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/azurestack/05_masters.json[]

8.5.16. ブートストラップの完了を待機し、Azure Stack Hub のブートストラップリソースを削除する

Microsoft Azure Stack Hub ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure Stack Hub で VNet および関連するサブネットを作成し、設定します。
- Azure Stack Hub でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ az network nsg rule delete -g ${RESOURCE_GROUP} --nsg-name ${INFRA_ID}-nsg --
name bootstrap_ssh_in
$ az vm stop -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm deallocate -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap --yes
$ az disk delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap_OSDisk --no-
wait --yes
$ az network nic delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-nic --no-
wait
$ az storage blob delete --account-key ${ACCOUNT_KEY} --account-name
${CLUSTER_NAME}sa --container-name files --name bootstrap.ign
$ az network public-ip delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-
ssh-pip
```

注記

ブートストラップサーバーを削除しないと、API トラフィックがブートストラップサーバーにルーティングされるため、インストールが成功しない場合があります。

8.5.17. Azure Stack Hub での追加のワーカーマシンの作成

Microsoft Azure Stack Hub でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Azure Resource Manager (ARM) テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_workers.json** というタイプのリソースを追加して起動することができます。

提供される ARM テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合は、インストールログで Red Hat サポートに接続することを検討してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure Stack Hub で VNet および関連するサブネットを作成し、設定します。
- Azure Stack Hub でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの **ワーカーマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **06_workers.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. ワーカーマシンのデプロイメントで必要な以下の変数をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/06_workers.json" \
  --parameters workerIgnition="${WORKER_IGNITION}" \ 1
  --parameters baseName="${INFRA_ID}" 2
  --parameters diagnosticsStorageAccountName="${CLUSTER_NAME}sa" 3
```

1 ワーカーノードの Ignition コンテンツ。

2 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3 クラスターのストレージアカウントの名前。

8.5.17.1. ワーカーマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例8.6 06_workers.json ARM テンプレート

link:https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/azurestack/06_workers.json[]

8.5.18. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

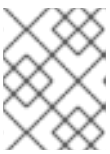
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

8.5.19. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

8.5.20. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

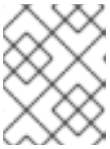
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

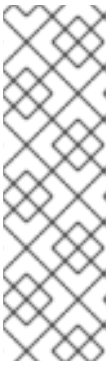
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

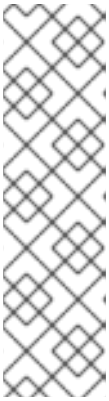
この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

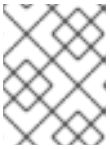
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

8.5.21. Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.*** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用して、OpenShift Container Platform クラスターを Microsoft Azure Stack Hub にデプロイしています。
- OpenShift CLI (**oc**) をインストールすること。
- Azure CLI のインストールまたは更新を実行します。

手順

1. Ingress ルーターがロードバランサーを作成し、**EXTERNAL-IP** フィールドにデータを設定していることを確認します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer   172.30.20.10    35.130.120.110
80:32288/TCP,443:31215/TCP   20
```

2. Ingress ルーター IP を変数としてエクスポートします。

```
$ export PUBLIC_IP_ROUTER=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

3. ***.apps** レコードを DNS ゾーンに追加します。

   a. このクラスターを新しい DNS ゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER} --ttl 300
```

   b. このクラスターを既存の DNS ゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n *.apps.${CLUSTER_NAME} -a ${PUBLIC_IP_ROUTER} --ttl 300
```

ワイルドカードを使用する代わりに明示的なドメインを追加する場合は、クラスターのそれぞれの現行ルートのエントリーを作成できます。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}
{end}' routes
```

出力例

```
oauth-openshift.apps.cluster.basedomain.com
console-openshift-console.apps.cluster.basedomain.com
downloads-openshift-console.apps.cluster.basedomain.com
alertmanager-main-openshift-monitoring.apps.cluster.basedomain.com
prometheus-k8s-openshift-monitoring.apps.cluster.basedomain.com
```

8.5.22. ユーザーによってプロビジョニングされるインフラストラクチャーでの Azure Stack Hub インストールの実行

Microsoft Azure Stack Hub のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる Azure Stack Hub インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

- クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

8.6. AZURE STACK HUB のインストール設定パラメーター

Azure Stack Hub の OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

8.6.1. Azure Stack Hub で使用できるインストール設定パラメーター

次の表では、インストールプロセスの一部として設定できる、必須、オプション、および Azure Stack Hub 固有のインストール設定パラメーターを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

8.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表8.15 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object

パラメーター	説明	値
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

8.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表8.16 ネットワークパラメーター

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

8.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表8.17 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列

パラメーター	説明	値
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**

パラメーター	説明	値
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。 **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**

パラメーター	説明	値
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。	**false** または **true**

パラメーター	説明	値
	重要  クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	

パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。  **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。

パラメーター	説明	値
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

8.6.1.4. 追加の AzureStackHub 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表8.18 追加の AzureStackHub パラメーター

パラメーター	説明	値
compute: platform: azure: osDisk: diskSizeGB:	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
compute: platform: azure: osDisk: diskType:	ディスクのタイプを定義します。	**standard_LRS** または **premium_LRS**。デフォルトは **premium_LRS** です。

パラメーター	説明	値
compute: platform: azure: type:	コンピュートマシンの azure インスタンスタイプを定義します。	String
controlPlane: platform: azure: osDisk: diskSizeGB:	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **1024** です。
controlPlane: platform: azure: osDisk: diskType:	ディスクのタイプを定義します。	**premium_LRS**.
controlPlane: platform: azure: type:	コントロールプレーンマシンの azure インスタンスタイプを定義します。	String
platform: azure: defaultMachinePlatform: osDisk: diskSizeGB:	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
platform: azure: defaultMachinePlatform: osDisk: diskType:	ディスクのタイプを定義します。	**standard_LRS** または **premium_LRS**。デフォルトは **premium_LRS** です。

パラメーター	説明	値
platform: azure: defaultMachinePlatform: type:	コントロールプレーンおよびコンピュートマシンの Azure インスタンスタイプ。	Azure インスタンスタイプ。
platform: azure: armEndpoint:	Azure Stack Hub Operator が提供する Azure Resource Manager エンドポイント の URL。	String
platform: azure: baseDomainResourceGroupName:	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform: azure: region:	Azure Stack Hub ローカルリージョンの名前。	String
platform: azure: resourceGroupName:	クラスターをインストールする既存のリソースグループの名前。このリソースグループは空で、この特定のクラスターにのみ使用する必要があります。クラスターコンポーネントは、リソースグループ内のすべてのリソースの所有権を想定します。インストールプログラムのサービスプリンシパルの範囲をこのリソースグループに制限する場合は、環境内でインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。	文字列 (例: **existing_resource_group**)。

パラメーター	説明	値
platform: azure: outboundType:	クラスターをインターネットに接続するために使用されるアウトバウンドルーティングストラテジー。ユーザー定義のルーティングを使用している場合、クラスターをインストールする前にアウトバウンドルーティングがすでに設定されている既存のネットワークが利用可能な状態にする必要があります。インストールプログラムはユーザー定義のルーティングの設定を行いません。	**LoadBalancer** または **UserDefinedRouting**。デフォルトは **LoadBalancer** です。
platform: azure: cloudName:	適切な Azure API エンドポイントで Azure SDK を設定するために使用される Azure クラウド環境の名前。	**AzureStackCloud**
clusterOSImage:	RHCOS VHD を含む Azure Stack 環境のストレージ blob の URL。	文字列 (たとえば、https://vhdsa.blob.example.example.com/vhd/rhcos-410.84.202112040202-0-azurestack.x86_64.vhd)

8.7. AZURE STACK HUB でのクラスターのアンインストール

Azure Stack Hub にデプロイしたクラスターを削除できます。

8.7.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第9章 GCP へのインストール

9.1. GCP へのインストールの準備

9.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。

9.1.2. OpenShift Container Platform の GCP へのインストール要件

OpenShift Container Platform を Google Cloud Platform (GCP) にインストールする前に、サービスアカウントを作成し、GCP プロジェクトを設定する必要があります。プロジェクトの作成、API サービスの有効化、DNS の設定、GCP アカウントの制限、およびサポート対象の GCP リージョンに関する詳細は、GCP プロジェクトの設定 を参照してください。

お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、他のオプションについて、GCP の長期間認証情報の手動作成 を参照してください。

9.1.3. GCP に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーによるプロビジョニングおよびユーザーによるプロビジョニングのインストールプロセスの詳細は、インストールプロセス を参照してください。

9.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる GCP インフラストラクチャーに、クラスターをインストールできます。

- **クラスターの GCP へのクイックインストール**: OpenShift Container Platform インストールプログラムでプロビジョニングされる GCP インフラストラクチャーに OpenShift Container Platform をインストールできます。デフォルトの設定オプションを使用して、クラスターを迅速にインストールできます。
- **カスタマイズされたクラスターの GCP へのインストール**: インストールプログラムがプロビジョニングする GCP インフラストラクチャーに、カスタマイズされたクラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の多くのカスタマイズオプションは、インストール後 に利用できます。
- **ネットワークのカスタマイズを使用したクラスターの GCP へのインストール**: インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。

- **ネットワークが制限された環境での GCP へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、インストーラーでプロビジョニングされる GCP インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。ミラーリングされたコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが GCP API を使用するにはインターネットへのアクセスが必要です。

- **クラスターの既存の Virtual Private Cloud へのインストール**: OpenShift Container Platform を既存の GCP Virtual Private Cloud (VPC) にインストールできます。このインストール方法は、新規アカウントまたはインフラストラクチャーを作成する際の制限など、会社のガイドラインによる制約がある場合に使用できます。

- **プライベートクラスターの既存の VPC へのインストール**: プライベートクラスターを既存の GCP VPC にインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

9.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、独自にプロビジョニングする GCP インフラストラクチャーにクラスターをインストールできます。

- **ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP へのクラスターのインストール**: 独自に提供する GCP インフラストラクチャーに OpenShift Container Platform をインストールできます。提供される Deployment Manager テンプレートを使用して、インストールを支援できます。

- **GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへの共有 VPC を設定したクラスターのインストール**: 提供される Deployment Manager テンプレートを使用して、共有 VPC インフラストラクチャーに GCP リソースを作成できます。

- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したネットワークが制限された環境での GCP へのクラスターのインストール**: ユーザーによってプロビジョニングされるインフラストラクチャーを使用して、ネットワークが制限された環境で GCP に OpenShift Container Platform をインストールできます。インストールリリースコンテンツの内部ミラーを作成することにより、ソフトウェアコンポーネントを取得するためのアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

9.1.4. 次のステップ

- GCP プロジェクトの設定

9.2. GCP プロジェクトの設定

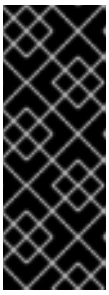
OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

9.2.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。



重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

9.2.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。インストールに不要なオプションの API サービスを有効にすることもできます。GCP ドキュメントの サービスの有効化 を参照してください。

表9.1 必要な API サービス

API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**

表9.2 オプションの API サービス

API サービス	コンソールサービス名
Google Cloud API	**cloudapis.googleapis.com**

API サービス	コンソールサービス名
Service Management API	**servicemanagement.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

9.2.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。

注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

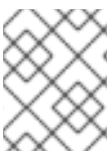
9.2.4. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表9.3 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	6	1
ファイアウォールのルール	Compute	グローバル	11	1
転送ルール	Compute	グローバル	2	0
使用中のグローバル IP アドレス	Compute	グローバル	4	1
ヘルスチェック	Compute	グローバル	3	0
イメージ	Compute	グローバル	1	0
ネットワーク	Compute	グローバル	2	0
静的 IP アドレス	Compute	リージョン	4	1
ルーター	Compute	グローバル	1	0
ルート	Compute	グローバル	2	0
サブネットワーク	Compute	グローバル	2	0
ターゲットプール	Compute	グローバル	3	0
CPU	Compute	リージョン	28	4
永続ディスク SSD (GB)	Compute	リージョン	896	128



注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

9.2.5. GCP でのサービスアカウントの作成

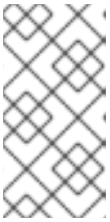
OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

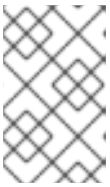
1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。



注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. サービスアカウントキーを JSON 形式で作成するか、サービスアカウントを GCP 仮想マシンにアタッチできます。GCP ドキュメントの サービスアカウントキー の 作成とインスタンスのサービスアカウントの作成と有効 化をご覧ください。
クラスターを作成するには、サービスアカウントキーまたはサービスアカウントがアタッチされた仮想マシンが必要です。



注記

サービスアカウントがアタッチされた仮想マシンを使用してクラスターを作成する場合は、インストール前に **install-config.yaml** ファイルで **credentialsMode: Manual** を設定する必要があります。

9.2.5.1. 必要な GCP のロール

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。組織のセキュリティーポリシーでより制限的なアクセス許可のセットが必要な場合は、次のアクセス許可を持つサービスアカウントを作成できます。クラスターを既存の VPC (virtual private cloud) にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- ロール管理者
- Security Admin
- Service Account Admin
- サービスアカウントキー管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS Administrator

Passthrough モードで Cloud Credential Operator を使用するために必要なロール

- ロードバランサー計算の管理者

次のロールは、コントロールプレーンとコンピュートマシンが使用するサービスアカウントに適用されます。

表9.4 GCP サービスアカウントのロール

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	**roles/compute.viewer**
	roles/storage.admin

9.2.5.2. インストーラーによってプロビジョニングされたインフラストラクチャーに必要な GCP 権限

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。

組織のセキュリティーポリシーで、より制限的なアクセス許可のセットが必要な場合は、必要なアクセス許可を持つ カスタムロール を作成できます。OpenShift Container Platform クラスターを作成および削除するために、インストーラーによってプロビジョニングされたインフラストラクチャーには、以下のパーミッションが必要です。

例9.1 ネットワークリソースの作成に必要な権限

- **compute.addresses.create**
- **compute.addresses.createInternal**
- **compute.addresses.delete**
- **compute.addresses.get**
- **compute.addresses.list**
- **compute.addresses.use**
- **compute.addresses.useInternal**
- **compute.firewalls.create**
- **compute.firewalls.delete**
- **compute.firewalls.get**
- **compute.firewalls.list**

- **compute.forwardingRules.create**
- **compute.forwardingRules.get**
- **compute.forwardingRules.list**
- **compute.forwardingRules.setLabels**
- **compute.networks.create**
- **compute.networks.get**
- **compute.networks.list**
- **compute.networks.updatePolicy**
- **compute.routers.create**
- **compute.routers.get**
- **compute.routers.list**
- **compute.routers.update**
- **compute.routes.list**
- **compute.subnetworks.create**
- **compute.subnetworks.get**
- **compute.subnetworks.list**
- **compute.subnetworks.use**
- **compute.subnetworks.useExternalIp**

例9.2 ロードバランサーリソースの作成に必要な権限

- **compute.regionBackendServices.create**
- **compute.regionBackendServices.get**
- **compute.regionBackendServices.list**
- **compute.regionBackendServices.update**
- **compute.regionBackendServices.use**
- **compute.targetPools.addInstance**
- **compute.targetPools.create**
- **compute.targetPools.get**
- **compute.targetPools.list**
- **compute.targetPools.removeInstance**

- **compute.targetPools.use**

例9.3 DNS リソースの作成に必要な権限

- **dns.changes.create**
- **dns.changes.get**
- **dns.managedZones.create**
- **dns.managedZones.get**
- **dns.managedZones.list**
- **dns.networks.bindPrivateDNSZone**
- **dns.resourceRecordSets.create**
- **dns.resourceRecordSets.list**

例9.4 サービスアカウントリソースの作成に必要な権限

- **iam.serviceAccountKeys.create**
- **iam.serviceAccountKeys.delete**
- **iam.serviceAccountKeys.get**
- **iam.serviceAccountKeys.list**
- **iam.serviceAccounts.actAs**
- **iam.serviceAccounts.create**
- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.get**
- **iam.serviceAccounts.list**
- **resourcemanager.projects.get**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**

例9.5 コンピューティングリソースの作成に必要な権限

- **compute.disks.create**
- **compute.disks.get**
- **compute.disks.list**

- **compute.disks.setLabels**
- **compute.instanceGroups.create**
- **compute.instanceGroups.delete**
- **compute.instanceGroups.get**
- **compute.instanceGroups.list**
- **compute.instanceGroups.update**
- **compute.instanceGroups.use**
- **compute.instances.create**
- **compute.instances.delete**
- **compute.instances.get**
- **compute.instances.list**
- **compute.instances.setLabels**
- **compute.instances.setMetadata**
- **compute.instances.setServiceAccount**
- **compute.instances.setTags**
- **compute.instances.use**
- **compute.machineTypes.get**
- **compute.machineTypes.list**

例9.6 ストレージリソースの作成に必要

- **storage.buckets.create**
- **storage.buckets.delete**
- **storage.buckets.get**
- **storage.buckets.list**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.get**
- **storage.objects.list**

例9.7 ヘルスチェックリソースを作成するために必要な権限

- **compute.healthChecks.create**
- **compute.healthChecks.get**
- **compute.healthChecks.list**
- **compute.healthChecks.useReadOnly**
- **compute.httpHealthChecks.create**
- **compute.httpHealthChecks.get**
- **compute.httpHealthChecks.list**
- **compute.httpHealthChecks.useReadOnly**

例9.8 GCP ゾーンとリージョン関連の情報を取得するために必要な権限

- **compute.globalOperations.get**
- **compute.regionOperations.get**
- **compute.regions.list**
- **compute.zoneOperations.get**
- **compute.zones.get**
- **compute.zones.list**

例9.9 サービスとクォータを確認するために必要な権限

- **monitoring.timeSeries.list**
- **serviceusage.quotas.get**
- **serviceusage.services.list**

例9.10 インストールに必要な IAM パーミッション

- **iam.roles.get**

例9.11 インストールのためのオプションのイメージ権限

- **compute.images.list**

例9.12 収集ブートストラップを実行するためのオプションの権限

- **compute.instances.getSerialPortOutput**

例9.13 ネットワークリソースを削除するために必要な権限

- compute.addresses.delete
- compute.addresses.deleteInternal
- compute.addresses.list
- compute.firewalls.delete
- compute.firewalls.list
- compute.forwardingRules.delete
- compute.forwardingRules.list
- compute.networks.delete
- compute.networks.list
- compute.networks.updatePolicy
- compute.routers.delete
- compute.routers.list
- compute.routes.list
- compute.subnetworks.delete
- compute.subnetworks.list

例9.14 ロードバランサーリソースを削除するために必要な権限

- compute.regionBackendServices.delete
- compute.regionBackendServices.list
- compute.targetPools.delete
- compute.targetPools.list

例9.15 DNS リソースを削除するために必要な権限

- dns.changes.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

例9.16 サービスアカウントリソースを削除するために必要な権限

- iam.serviceAccounts.delete
- iam.serviceAccounts.get
- iam.serviceAccounts.list
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy

例9.17 コンピューティングリソースを削除するために必要な権限

- compute.disks.delete
- compute.disks.list
- compute.instanceGroups.delete
- compute.instanceGroups.list
- compute.instances.delete
- compute.instances.list
- compute.instances.stop
- compute.machineTypes.list

例9.18 ストレージリソースの削除に必要

- storage.buckets.delete
- storage.buckets.getIamPolicy
- storage.buckets.list
- storage.objects.delete
- storage.objects.list

例9.19 ヘルスチェックリソースを削除するために必要な権限

- compute.healthChecks.delete
- compute.healthChecks.list
- compute.httpHealthChecks.delete
- compute.httpHealthChecks.list

-

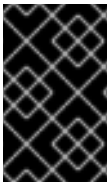
例9.20 削除に必要なイメージ権限

- **compute.images.list**

9.2.5.3. 共有 VPC インストールに必要な GCP パーミッション

クラスターを 共有 VPC にインストールする場合は、ホストプロジェクトとサービスプロジェクトの両方のサービスアカウントを設定する必要があります。共有 VPC にインストールしない場合は、このセクションをスキップできます。

上記の標準インストールに必要な最小限のロールをサービスプロジェクトに適用する必要があります。



重要

手動または mint 認証情報モードのいずれかで動作する Cloud Credential Operator に粒度の細かいパーミッションを使用できます。パススルー認証情報モードでは、粒度の細かいパーミッションを使用することはできません。

ホストプロジェクトが以下の設定のいずれかをサービスアカウントに適用することを確認します。

例9.21 ホストプロジェクトでファイアウォールを作成するために必要な権限

- **projects/<host-project>/roles/dns.networks.bindPrivateDNSZone**
- **roles/compute.networkAdmin**
- **roles/compute.securityAdmin**

例9.22 必要な最小限の権限

- **projects/<host-project>/roles/dns.networks.bindPrivateDNSZone**
- **roles/compute.networkUser**

9.2.6. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **africa-south1** (Johannesburg, South Africa)
- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)

- **asia-south1** (Mumbai, India)
- **asia-south2** (Delhi, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **australia-southeast2** (Melbourne, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-southwest1** (Madrid, Spain)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **europe-west8** (Milan, Italy)
- **europe-west9** (Paris, France)
- **europe-west12** (Turin, Italy)
- **me-central1** (Doha, Qatar, Middle East)
- **me-central2** (Dammam, Saudi Arabia, Middle East)
- **me-west1** (Tel Aviv, Israel)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **northamerica-northeast2** (Toronto, Ontario, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **southamerica-west1** (Santiago, Chile)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-east5** (Columbus, Ohio)
- **us-south1** (Dallas, Texas)

- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)

注記

リージョンおよびゾーンごとにどのマシンタイプのインスタンスが使用できるかを確認するには、Google の ドキュメント を参照してください。

9.2.7. 次のステップ

- GCP に OpenShift Container Platform クラスターをインストールします。カスタマイズされたクラスターのインストール、またはデフォルトのオプションで クラスターのクイックインストール を実行できます。

9.3. GCP へのクラスターのクイックインストール

OpenShift Container Platform バージョン 4.16 では、デフォルトの設定オプションを使用するクラスターを Google Cloud Platform (GCP) にインストールできます。

9.3.1. 前提条件

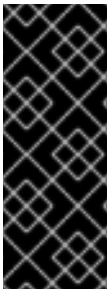
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

9.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.3.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

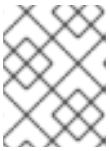
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.3.4. インストールプログラムの取得

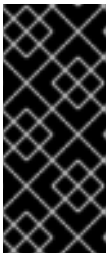
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

9.3.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

   ディレクトリーを指定する場合:
   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。
3. プロンプト時に値を指定します。
   a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **gcp** を選択します。

c. ホスト上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、ファイルへの絶対パスを入力する必要があります。

d. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

e. クラスターをデプロイするリージョンを選択します。

f. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

g. クラスターの記述名を入力します。7 文字以上の名前を指定すると、クラスター名から生成されるインフラストラクチャー ID で最初の 6 文字のみが使用されます。

h. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

4. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。

- **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
- **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
```

```
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

9.3.6. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

   ```
   $ tar xvf <file>
   ```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
   PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。

5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.3.7. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

9.3.8. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に

実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.3.9. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

9.4. カスタマイズによる GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが Google Cloud Platform (GCP) 上にプロビジョニングするインフラストラクチャーに、カスタマイズしたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

9.4.1. 前提条件

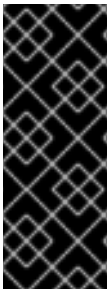
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

9.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.4.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

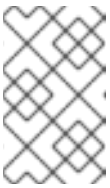
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.4.4. インストールプログラムの取得

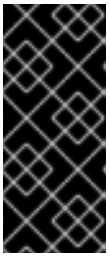
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

9.4.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、ファイルへの絶対パスを入力する必要があります。

      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

      v. クラスターをデプロイするリージョンを選択します。

      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

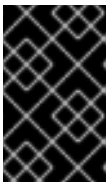
      vii. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

注記

3 ノードクラスターをインストールする場合は、必ず **compute.replicas** パラメーターを **0** に設定してください。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細については、「GCP に 3 ノードクラスターをインストールする」を参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- GCP のインストール設定パラメーター

9.4.5.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表9.5 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と

保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

9.4.5.2. GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform インスタンスタイプは OpenShift Container Platform でテストされています。

例9.23 マシンのシリーズ

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**
- **E2**
- **M1**
- **N1**
- **N2**
- **N2D**

- **Tau T2D**

9.4.5.3. 64 ビット ARM インフラストラクチャー上の GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform (GCP) 64 ビット ARM インスタンスタイプは OpenShift Container Platform でテストされています。

例9.24 64 ビット ARM マシン用のマシンシリーズ

- **Tau T2A**

9.4.5.4. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプは、コントロールプレーンとコンピューティングマシンの最小リソース要件を満たす必要があります。詳細については、「クラスターインストールの最小リソース要件」を参照してください。
- カスタムマシンタイプの名前は、次の構文に従う必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプのサンプル install-config.yaml ファイル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

9.4.5.5. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。

注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

  b. コンピューティングマシンのみに Shielded VM を使用するには:

```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

  c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.4.5.6. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中のデータを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。

注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled 1
      type: n2d-standard-8 2
      onHostMaintenance: Terminate 3
```

1 Confidential VM を有効にします。

2 Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタイプの詳細については、サポートされているオペレーティングシステムとマシンタイプ を参照してください。

3 ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

  b. コンピューティングマシンのみに Confidential VM を使用するには:

```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

  c. すべてのマシンに Confidential VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.4.5.7. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (6)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
      tags: (7)
      - control-plane-tag1
      - control-plane-tag2
      osImage: (8)
        project: example-project-name
        name: example-image-name
  replicas: 3
compute: (9) (10)
- hyperthreading: Enabled (11)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (12)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
        tags: (13)
        - compute-tag1
        - compute-tag2
        osImage: (14)
          project: example-project-name
          name: example-image-name
  replicas: 3
metadata:
```

```
  name: test-cluster 15
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 16
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 17
    region: us-central1 18
    defaultMachinePlatform:
      tags: 19
      - global-tag1
      - global-tag2
      osImage: 20
        project: example-project-name
        name: example-image-name
pullSecret: '{"auths": ...}' 21
fips: false 22
sshKey: ssh-ed25519 AAAA... 23
```

1 15 17 18 21 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 9 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 10 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 11 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

6 12 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカ

ウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントに適切な権限を付与する方法の詳細については、マシン管理 → コンピュートマシンセットの作成→ GCP でのコンピューティングマシンセットの作成を参照してください。

7 13 19 オプション: コントロールプレーンまたはコンピューティングマシンセットに適用するネットワークタグのセット。**platform.gcp.defaultMachinePlatform.tags** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**compute.platform.gcp.tags** パラメーターまたは **controlPlane.platform.gcp.tags** パラメーターが設定されている場合は、**platform.gcp.defaultMachinePlatform.tags** パラメーターを上書きします。

8 14 20 オプション: コントロールプレーンとコンピュートマシンの起動に使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS)。**platform.gcp.defaultMachinePlatform.osImage** の下の **project** および **name** パラメーターは、コントロールプレーンマシンとコンピュートマシンの両方に適用されます。**controlPlane.platform.gcp.osImage** または **compute.platform.gcp.osImage** の下の **project** および **name** パラメーターが設定されている場合、それらは **platform.gcp.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

16 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

22 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

23 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

関連情報

- コンピュートマシンセットの顧客管理の暗号鍵の有効化

9.4.5.8. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

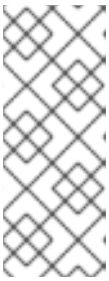
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5.

オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** で

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



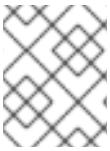
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.4.6. GCP のユーザー定義のラベルとタグの管理

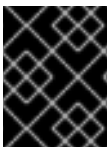
重要

GCP のユーザー定義のラベルとタグのサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

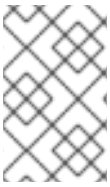
Google Cloud Platform (GCP) は、特定の OpenShift Container Platform クラスター用に作成されたリソースの識別と整理に役立つラベルとタグを提供し、管理を容易にします。

各 GCP リソースのラベルとタグは、OpenShift Container Platform クラスターのインストール中にのみ定義できます。



重要

ユーザー定義のラベルとタグは、OpenShift Container Platform 4.16 にアップグレードされた OpenShift Container Platform クラスターではサポートされません。



注記

すでに追加されているタグを更新することはできません。また、設定されたタグキーまたはタグ値が削除された場合、タグによってサポートされるリソースの新規作成が失敗します。

ユーザー定義のラベル

ユーザー定義のラベルと OpenShift Container Platform 固有のラベルは、OpenShift Container Platform インストールプログラムとそのコアコンポーネントによって作成されたリソースにのみ適用されます。

- GCP ファイルストアの CSI ドライバーオペレーター
- GCP PD CSI Driver Operator
- Image Registry Operator
- GCP 用のマシン API プロバイダー

ユーザー定義のタグは、OpenShift Container Platform インストールプログラム、Image Registry Operator、および Machine API Operator によって作成されたリソースに割り当てられます。ユーザー定義のタグは、他の Operator または Kubernetes のツリー内コンポーネントによって作成されたリソースには割り当てられません。

ユーザー定義のラベルと OpenShift Container Platform ラベルは、次の GCP リソースで使用できます。

- コンピュートディスク
- コンピュートインスタンス
- コンピュートイメージ
- コンピュート転送ルール
- DNS 管理ゾーン
- ファイルストアインスタンス
- ストレージバケット

ユーザー定義ラベルの制限

- **ComputeAddress** のラベルは GCP ベータ版でサポートされています。OpenShift Container Platform はリソースにラベルを追加しません。

ユーザー定義のタグ

ユーザー定義のタグは、OpenShift Container Platform Image Registry Operator によって作成されたリソースに付加され、他の Operator または Kubernetes ツリー内コンポーネントによって作成されたリソースには付加されません。

ユーザー定義のタグは、次の GCP リソースで利用できます。

- ストレージバケット

- コンピュートインスタンス
- コンピュートディスク

ユーザー定義タグの制限事項

- 以下の商品にはタグはつきません。
  - GCP ファイルストア CSI ドライバーオペレーターによって作成されたファイルストアインスタンスリソース
  - GCP PD CSI ドライバーオペレーターによって作成されたコンピュートディスクとコンピュートイメージリソース
- Operator は最小限のロールを持つサービスアカウントを作成して使用するため、タグを特定のサービスアカウントに制限してはなりません。
- OpenShift Container Platform は、タグのキーおよび値のリソースを作成しません。
- OpenShift Container Platform 固有のタグはどのリソースにも追加されません。

関連情報

- **OrganizationID** の識別の詳細は、OrganizationID を参照してください。
- **ProjectID** の識別の詳細は、ProjectID を参照してください。
- ラベルの詳細は、ラベルの概要 を参照してください。
- タグの詳細は、Tags Overview を参照してください。

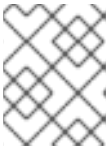
9.4.6.1. GCP のユーザー定義のラベルとタグの設定

前提条件

- インストールプログラムでは、組織レベルとプロジェクトレベルの両方で定義されたタグを使用してプログラムが OpenShift Container Platform クラスターを作成できるように、サービスアカウントに **TagUser** ロールが含まれている必要があります。

手順

- **install-config.yaml** ファイルを更新して、必要なラベルとタグのリストを定義します。



注記

ラベルとタグは **install-config.yaml** の作成フェーズ中に定義され、クラスターの作成後に新しいラベルやタグで変更または更新することはできません。

install-config.yaml ファイルのサンプル

```
apiVersion: v1
featureSet: TechPreviewNoUpgrade
platform:
 gcp:
```

```
userLabels: 1
- key: <label_key>2
  value: <label_value>3
userTags: 4
- parentID: <OrganizationID/ProjectID>5
  key: <tag_key_short_name>
  value: <tag_value_short_name>
```

1 GCP で作成されたリソースにキーと値をラベルとして追加します。

2 ラベル名を定義します。

3 ラベルの内容を定義します。

4 GCP 上で作成したリソースにキーと値をタグとして追加します。

5 組織またはプロジェクトレベルで、タグが定義されている階層リソースの ID。

ユーザー定義ラベルの要件は次のとおりです。

- ラベルのキーと値は、最小 1 文字、最大 63 文字である必要があります。
- ラベルのキーと値には、小文字、数字、アンダースコア (_)、およびダッシュ (-) のみを含める必要があります。
- ラベルキーは小文字で始まる必要があります。
- リソースごとに最大 32 個のラベルを設定できます。各リソースには最大 64 個のラベルを含めることができ、32 個のラベルは OpenShift Container Platform による内部使用のために予約されています。

ユーザー定義タグの要件は次のとおりです。

- タグキーとタグ値はすでに存在している必要があります。OpenShift Container Platform はキーと値を作成しません。
- タグの **parentID** は、**OrganizationID** または **ProjectID** のいずれかになります。
  - **OrganizationID** は、先行ゼロのない 10 進数で設定されている必要があります。
  - **ProjectID** の長さは 6 ~ 30 文字で、小文字、数字、ハイフンのみを含む必要があります。
  - **ProjectID** は文字で始まる必要があり、ハイフンで終わることはできません。
- タグキーには、大文字と小文字の英数字、ハイフン (-)、アンダースコア (_)、およびピリオド (.) のみを含める必要があります。
- タグ値には、大文字と小文字の英数字、ハイフン (-)、アンダースコア (_)、ピリオド (.)、アットマーク (@)、パーセント記号 (%)、等号 (=)、プラス記号 (+)、コロン (:)、コンマ (,)、アスタリスク (*)、ポンド記号 (#)、アンパサンド (&)、括弧 (())、角中括弧 ([])、中括弧 ({})、およびスペースのみを含める必要があります。
- タグのキーと値は英数字で始まり、終わる必要があります。
- タグ値は、キーの事前定義された値の 1 つである必要があります。

- 最大 50 個のタグを設定できます。
- 親リソースから継承される既存のタグキーと同じ値で定義されたタグキーが存在してはなりません。

9.4.6.2. GCP のユーザー定義のラベルとタグのクエリー

OpenShift Container Platform クラスターを作成した後、次のサンプル **infrastructure.yaml** ファイルに示すように、**infrastructures.config.openshift.io/cluster** オブジェクト内の GCP リソースに定義されたラベルとタグのリストにアクセスできます。

サンプルの infrastructure.yaml ファイル

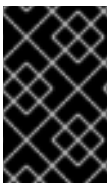
```
apiVersion: config.openshift.io/v1
kind: Infrastructure
metadata:
 name: cluster
spec:
 platformSpec:
   type: GCP
status:
 infrastructureName: <cluster_id> 1
 platform: GCP
 platformStatus:
   gcp:
     resourceLabels:
     - key: <label_key>
       value: <label_value>
     resourceTags:
     - key: <tag_key_short_name>
       parentID: <OrganizationID/ProjectID>
       value: <tag_value_short_name>
   type: GCP
```

1 クラスターのインストール中に生成されるクラスター ID。

ユーザー定義のラベルに加えて、リソースには OpenShift Container Platform によって定義されたラベルがあります。OpenShift Container Platform ラベルの形式は、**kubernetes-io-cluster-<cluster_id>:owned** です。

9.4.7. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

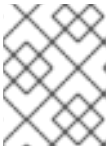
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.4.8. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように GCP クラスターを設定する の手順に従ってください。

9.4.8.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

   例9.25 必要な GCP パーミッション

   - compute.machineTypes.list
   - compute.regions.list
   - compute.zones.list
   - dns.changes.create
   - dns.changes.get
   - dns.managedZones.create
   - dns.managedZones.delete
   - dns.managedZones.get
   - dns.managedZones.list
   - dns.networks.bindPrivateDNSZone
   - dns.resourceRecordSets.create
   - dns.resourceRecordSets.delete
   - dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

5. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   ```

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
    ...
```

6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

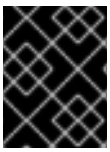
シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
      ...
  secretRef:
```

```
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```



重要

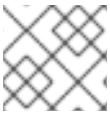
手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

9.4.8.2. 短期認証情報を使用するように GCP クラスターを設定

GCP Workload Identity を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な GCP リソースを作成する必要があります。

9.4.8.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- インストールプログラムが使用する GCP アカウントに次のいずれかの認証方法を追加している。
  - **IAM Workload Identity Pool Admin**ロール
  - 次の詳細な権限:

    例9.26 必要な GCP パーミッション

    - compute.projects.get
    - iam.googleapis.com/workloadIdentityPoolProviders.create
    - iam.googleapis.com/workloadIdentityPoolProviders.get

- iam.googleapis.com/workloadIdentityPools.create
- iam.googleapis.com/workloadIdentityPools.delete
- iam.googleapis.com/workloadIdentityPools.get
- iam.googleapis.com/workloadIdentityPools.undelete
- iam.roles.create
- iam.roles.delete
- iam.roles.list
- iam.roles.undelete
- iam.roles.update
- iam.serviceAccounts.create
- iam.serviceAccounts.delete
- iam.serviceAccounts.getIamPolicy
- iam.serviceAccounts.list
- iam.serviceAccounts.setIamPolicy
- iam.workloadIdentityPoolProviders.get
- iam.workloadIdentityPools.delete
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy
- storage.buckets.create
- storage.buckets.delete
- storage.buckets.get
- storage.buckets.getIamPolicy
- storage.buckets.setIamPolicy
- storage.objects.create
- storage.objects.delete
- storage.objects.list

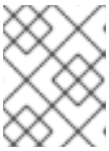
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

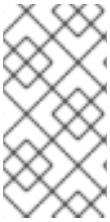
ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
```

```
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

9.4.8.2.2. Cloud Credential Operator ユーティリティーを使用した GCP リソースの作成

ccoctl gcp create-all コマンドを使用して、GCP リソースの作成を自動化できます。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

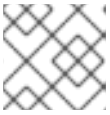
1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。



注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl gcp create-all \
  --name=<name> \1
  --region=<gcp_region> \2
  --project=<gcp_project_id> \3
  --credentials-requests-dir=<path_to_credentials_requests_directory> 4
```

1. トラッキングに使用される、作成されたすべての GCP リソースのユーザー定義名を指定します。
2. クラウドリソースを作成する GCP リージョンを指定します。
3. クラウドリソースを作成する GCP プロジェクト ID を指定します。
4. GCP サービスアカウントを作成するには、**CredentialsRequest** マニフェストのファイルが含まれるディレクトリーを指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

GCP にクエリーを実行すると、IAM サービスアカウントが作成されていることを確認できます。詳細については、IAM サービスアカウントのリスト表示に関する GCP のドキュメントを参照してください。

9.4.8.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

   例9.27 必要な GCP パーミッション

   - compute.machineTypes.list
   - compute.regions.list
   - compute.zones.list
   - dns.changes.create
   - dns.changes.get
   - dns.managedZones.create
   - dns.managedZones.delete
   - dns.managedZones.get
   - dns.managedZones.list
   - dns.networks.bindPrivateDNSZone
   - dns.resourceRecordSets.create
   - dns.resourceRecordSets.delete
   - dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

9.4.9. GCP Marketplace の使用

GCP Marketplace を使用すると、OpenShift Container Platform クラスターをデプロイできます。これは、GCP を通じて従量課金制 (時間単位、コア単位) で請求され、Red Hat の直接サポートも受けることができます。

デフォルトで、インストールプログラムはコンピュートマシンのデプロイに使用する Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードしてインストールします。GCP Marketplace から RHCOS イメージを使用して OpenShift Container Platform クラスターをデプロイするには、GCP Marketplace サービの場所を参照するように **install-config.yaml** ファイルを変更してデフォルトの動作をオーバーライドします。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

1. **compute.platform.gcp.osImage** パラメーターを編集して、GCP Marketplace イメージの場所を指定します。
   - **project** パラメーターを **redhat-marketplace-public** に設定します。
   - **name** パラメーターを、次のいずれかに設定します。

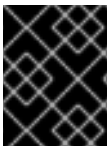
     OpenShift Container Platform
     redhat-coreos-ocp-413-x86-64-202305021736
     OpenShift Platform Plus
     redhat-coreos-opp-413-x86-64-202305021736
     OpenShift Kubernetes Engine
     redhat-coreos-oke-413-x86-64-202305021736
2. ファイルを保存し、クラスターをデプロイする際に参照します。

コンピュートマシンの GCP Marketplace イメージを指定するサンプル install-config.yaml ファイル

```
apiVersion: v1
baseDomain: example.com
controlPlane:
# ...
compute:
  platform:
    gcp:
      osImage:
        project: redhat-marketplace-public
        name: redhat-coreos-ocp-413-x86-64-202305021736
# ...
```

9.4.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

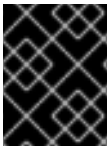
1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

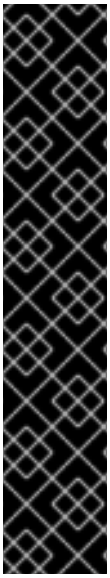


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

9.4.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

9.4.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.4.13. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

9.5. ネットワークのカスタマイズによる GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが Google Cloud Platform (GCP) 上にプロビジョニングするインフラストラクチャーに、カスタマイズしたネットワーク設定でクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

9.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

9.5.2. OpenShift Container Platform のインターネットアクセス

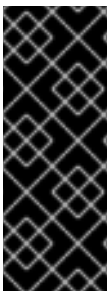
OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を

無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.5.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

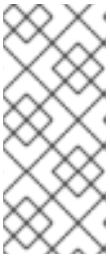
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

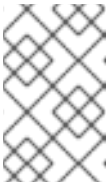
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.5.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

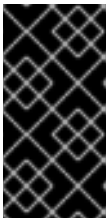
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

9.5.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。
      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、ファイルへの絶対パスを入力する必要があります。
      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。
      v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- GCP のインストール設定パラメーター

9.5.5.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表9.6 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と

保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

9.5.5.2. GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform インスタンスタイプは OpenShift Container Platform でテストされています。

例9.28 マシンのシリーズ

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**
- **E2**
- **M1**
- **N1**
- **N2**
- **N2D**

- **Tau T2D**

9.5.5.3. 64 ビット ARM インフラストラクチャー上の GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform (GCP) 64 ビット ARM インスタンスタイプは OpenShift Container Platform でテストされています。

例9.29 64 ビット ARM マシン用のマシンシリーズ

- **Tau T2A**

9.5.5.4. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプは、コントロールプレーンとコンピューティングマシンの最小リソース要件を満たす必要があります。詳細については、「クラスターインストールの最小リソース要件」を参照してください。
- カスタムマシンタイプの名前は、次の構文に従う必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

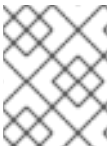
インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプのサンプル install-config.yaml ファイル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

9.5.5.5. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。



注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

  b. コンピューティングマシンのみに Shielded VM を使用するには:

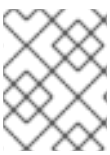
```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

  c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.5.5.6. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中のデータを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。



注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled 1
      type: n2d-standard-8 2
      onHostMaintenance: Terminate 3
```

1 Confidential VM を有効にします。

2 Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタイプの詳細については、サポートされているオペレーティングシステムとマシンタイプ を参照してください。

3 ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

  b. コンピューティングマシンのみに Confidential VM を使用するには:

```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

  c. すべてのマシンに Confidential VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.5.5.7. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (6)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
      tags: (7)
      - control-plane-tag1
      - control-plane-tag2
      osImage: (8)
        project: example-project-name
        name: example-image-name
  replicas: 3
compute: (9) (10)
- hyperthreading: Enabled (11)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (12)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
        tags: (13)
        - compute-tag1
        - compute-tag2
        osImage: (14)
          project: example-project-name
          name: example-image-name
  replicas: 3
metadata:
```

```
  name: test-cluster 15
networking: 16
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 17
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 18
    region: us-central1 19
    defaultMachinePlatform:
      tags: 20
      - global-tag1
      - global-tag2
      osImage: 21
        project: example-project-name
        name: example-image-name
pullSecret: '{"auths": ...}' 22
fips: false 23
sshKey: ssh-ed25519 AAAA... 24
```

1 15 18 19 22 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 9 16 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 10 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 11 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

6 12 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカ

待つ役割、適切な権限が割り当てられている必要があります。デフォルトのサービスアカウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントに適切な権限を付与する方法の詳細については、マシン管理 → コンピュートマシンセットの作成→ GCP でのコンピューティングマシンセットの作成を参照してください。

7 13 20 オプション: コントロールプレーンまたはコンピューティングマシンセットに適用するネットワークタグのセット。**platform.gcp.defaultMachinePlatform.tags** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**compute.platform.gcp.tags** パラメーターまたは **controlPlane.platform.gcp.tags** パラメーターが設定されている場合は、**platform.gcp.defaultMachinePlatform.tags** パラメーターを上書きします。

8 14 21 オプション: コントロールプレーンとコンピュートマシンの起動に使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS)。**platform.gcp.defaultMachinePlatform.osImage** の下の **project** および **name** パラメーターは、コントロールプレーンマシンとコンピュートマシンの両方に適用されます。**controlPlane.platform.gcp.osImage** または **compute.platform.gcp.osImage** の下の **project** および **name** パラメーターが設定されている場合、それらは **platform.gcp.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

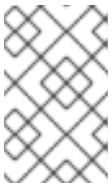
17 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

23 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

24 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

関連情報

- コンピュートマシンセットの顧客管理の暗号鍵の有効化

9.5.5.8. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

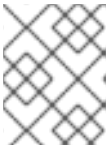
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

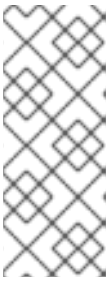
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5.

オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** で



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

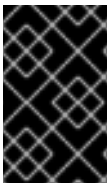
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.5.6. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。

5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.5.7. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。

- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように GCP クラスターを設定する の手順に従ってください。

9.5.7.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

例9.30 必要な GCP パーミッション

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get

- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。

3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
  ...
```

6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
```

```
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```

重要

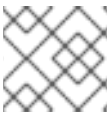
手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

9.5.7.2. 短期認証情報を使用するように GCP クラスターを設定

GCP Workload Identity を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な GCP リソースを作成する必要があります。

9.5.7.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- インストールプログラムが使用する GCP アカウントに次のいずれかの認証方法を追加している。
  - **IAM Workload Identity Pool Admin**ロール
  - 次の詳細な権限:

    例9.31 必要な GCP パーミッション

    - compute.projects.get
    - iam.googleapis.com/workloadIdentityPoolProviders.create
    - iam.googleapis.com/workloadIdentityPoolProviders.get
    - iam.googleapis.com/workloadIdentityPools.create
    - iam.googleapis.com/workloadIdentityPools.delete
    - iam.googleapis.com/workloadIdentityPools.get
    - iam.googleapis.com/workloadIdentityPools.undelete
    - iam.roles.create
    - iam.roles.delete

- iam.roles.list
- iam.roles.undelete
- iam.roles.update
- iam.serviceAccounts.create
- iam.serviceAccounts.delete
- iam.serviceAccounts.getIamPolicy
- iam.serviceAccounts.list
- iam.serviceAccounts.setIamPolicy
- iam.workloadIdentityPoolProviders.get
- iam.workloadIdentityPools.delete
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy
- storage.buckets.create
- storage.buckets.delete
- storage.buckets.get
- storage.buckets.getIamPolicy
- storage.buckets.setIamPolicy
- storage.objects.create
- storage.objects.delete
- storage.objects.list

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

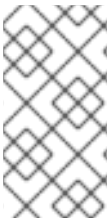
```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

9.5.7.2.2. Cloud Credential Operator ユーティリティーを使用した GCP リソースの作成

ccoctl gcp create-all コマンドを使用して、GCP リソースの作成を自動化できます。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl gcp create-all \
  --name=<name> \ 1
  --region=<gcp_region> \ 2
  --project=<gcp_project_id> \ 3
  --credentials-requests-dir=<path_to_credentials_requests_directory> 4
```

1. トラッキングに使用される、作成されたすべての GCP リソースのユーザー定義名を指定します。
2. クラウドリソースを作成する GCP リージョンを指定します。
3. クラウドリソースを作成する GCP プロジェクト ID を指定します。
4. GCP サービスアカウントを作成するには、**CredentialsRequest** マニフェストのファイルが含まれるディレクトリーを指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

GCP にクエリーを実行すると、IAM サービスアカウントが作成されていることを確認できます。詳細については、IAM サービスアカウントのリスト表示に関する GCP のドキュメントを参照してください。

9.5.7.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

   例9.32 必要な GCP パーミッション

   - compute.machineTypes.list
   - compute.regions.list
   - compute.zones.list
   - dns.changes.create
   - dns.changes.get
   - dns.managedZones.create
   - dns.managedZones.delete
   - dns.managedZones.get
   - dns.managedZones.list
   - dns.networks.bindPrivateDNSZone
   - dns.resourceRecordSets.create
   - dns.resourceRecordSets.delete
   - dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

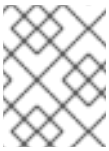
9.5.8. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

重要

CIDR 範囲 **172.17.0.0/16** は libVirt によって予約されています。この範囲、またはこの範囲と重複する範囲をクラスター内のネットワークに使用することはできません。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 でネットワークプラグインをさらにカスタマイズできます。

9.5.9. 高度なネットワーク設定の指定

ネットワークプラグインに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 次の例のように、**cluster-network-03-config.yml** ファイルでクラスターの高度なネットワーク設定を指定します。

   OVN-Kubernetes ネットワークプロバイダーを有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

9.5.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

9.5.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表9.7 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。

フィールド	型	説明
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表9.8 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表9.9 ovnKubernetesConfig オブジェクト

フィールド	型	説明

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。  **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表9.10 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表9.11 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表9.12 **policyAuditConfig オブジェクト**

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表9.13 **gatewayConfig** オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

フィールド	型	説明
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表9.14 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表9.15 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表9.16 **ipsecConfig オブジェクト**

フィールド	型	説明

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

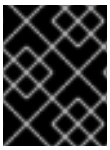
表9.17 **kubeProxyConfig** オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	型	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

9.5.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

9.5.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

9.5.13. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.5.14. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

9.6. ネットワークが制限された環境での GCP へのクラスターのインストール

OpenShift Container Platform 4.16 では、既存の Google Virtual Private Cloud (VPC) にインストールリリースコンテンツの内部ミラーを作成することで、制限されたネットワーク内の Google Cloud Platform (GCP) にクラスターをインストールできます。

重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが GCP API を使用するにはインターネットアクセスが必要になります。

9.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように GCP プロジェクトを設定 している。
- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

  重要

  インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- GCP に既存の VPC がある。インストーラーでプロビジョニングされるインフラストラクチャーを使用するネットワークが制限された環境にクラスターをインストールする場合は、イ

ンストーラーでプロビジョニングされる VPC を使用することはできません。以下の要件のいずれかを満たすユーザーによってプロビジョニングされる VPC を使用する必要があります。

- ミラーレジストリーが含まれる。
- 別の場所でホストされるミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続がある。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。他のサイトへのアクセスを付与する必要がある場合もありますが、***.googleapis.com** および **accounts.google.com** へのアクセスを付与する必要があります。

9.6.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

9.6.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

9.6.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

9.6.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

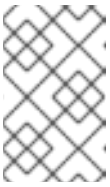
一部のディストリビューションでは、**~/.ssh/id_rsa** および **~/.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.6.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

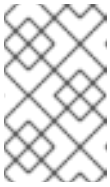
```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。
      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、ファイルへの絶対パスを入力する必要があります。
      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。
      v. クラスターをデプロイするリージョンを選択します。
      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。
      vii. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

   a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

   ```
   pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
   "you@example.com"}}}'
   ```

   <mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

   b. **additionalTrustBundle** パラメーターおよび値を追加します。

   ```
   additionalTrustBundle: |
     -----BEGIN CERTIFICATE-----

   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
     -----END CERTIFICATE-----
   ```

   この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

   c. VPC のネットワークおよびサブネットを定義して、親の **platform.gcp** フィールドの下にクラスターをインストールします。

   ```
   network: <existing_vpc>
   controlPlaneSubnet: <control_plane_subnet>
   computeSubnet: <compute_subnet>
   ```

   platform.gcp.network には、既存の Google VPC の名前を指定します。**platform.gcp.controlPlaneSubnet** および **platform.gcp.computeSubnet** の場合には、コントロールプレーンマシンとコンピュートマシンをそれぞれデプロイするために既存のサブネットを指定します。

   d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

   ```
   imageContentSources:
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: quay.io/openshift-release-dev/ocp-release
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: registry.redhat.io/ocp/release
   ```

   これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

   e. オプション: パブリッシュストラテジーを **Internal** に設定します。

   ```
   publish: Internal
   ```

   このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- GCP のインストール設定パラメーター

9.6.5.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表9.18 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

9.6.5.2. GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform インスタンスタイプは OpenShift Container Platform でテストされています。

例9.33 マシンのシリーズ

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**
- **E2**
- **M1**
- **N1**
- **N2**
- **N2D**
- **Tau T2D**

9.6.5.3. 64 ビット ARM インフラストラクチャー上の GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform (GCP) 64 ビット ARM インスタンスタイプは OpenShift Container Platform でテストされています。

例9.34 64 ビット ARM マシン用のマシンシリーズ

- **Tau T2A**

9.6.5.4. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプは、コントロールプレーンとコンピューティングマシンの最小リソース要件を満たす必要があります。詳細については、「クラスターインストールの最小リソース要件」を参照してください。
- カスタムマシンタイプの名前は、次の構文に従う必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプのサンプル install-config.yaml ファイル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

9.6.5.5. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。

注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

  b. コンピューティングマシンのみに Shielded VM を使用するには:

```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

  c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.6.5.6. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中のデータを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。

注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled 1
      type: n2d-standard-8 2
      onHostMaintenance: Terminate 3
```

1 Confidential VM を有効にします。

2 Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタイプの詳細については、サポートされているオペレーティングシステムとマシンタイプ を参照してください。

3 ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

  b. コンピューティングマシンのみに Confidential VM を使用するには:

```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

  c. すべてのマシンに Confidential VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.6.5.7. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
```

```
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (6)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
      tags: (7)
      - control-plane-tag1
      - control-plane-tag2
      osImage: (8)
        project: example-project-name
        name: example-image-name
  replicas: 3
compute: (9) (10)
- hyperthreading: Enabled (11)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (12)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
        tags: (13)
        - compute-tag1
        - compute-tag2
        osImage: (14)
          project: example-project-name
          name: example-image-name
  replicas: 3
metadata:
  name: test-cluster (15)
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 16
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 17
    region: us-central1 18
    defaultMachinePlatform:
      tags: 19
      - global-tag1
      - global-tag2
      osImage: 20
        project: example-project-name
        name: example-image-name
    network: existing_vpc 21
    controlPlaneSubnet: control_plane_subnet 22
    computeSubnet: compute_subnet 23
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 24
fips: false 25
sshKey: ssh-ed25519 AAAA... 26
additionalTrustBundle: | 27
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: 28
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 15 17 18 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 9 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 10 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 11

同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

6 12 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントに適切な権限を付与する方法の詳細については、マシン管理 → コンピュートマシンセットの作成→ GCP でのコンピューティングマシンセットの作成を参照してください。

7 13 19 オプション: コントロールプレーンまたはコンピューティングマシンセットに適用するネットワークタグのセット。**platform.gcp.defaultMachinePlatform.tags** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**compute.platform.gcp.tags** パラメーターまたは **controlPlane.platform.gcp.tags** パラメーターが設定されている場合は、**platform.gcp.defaultMachinePlatform.tags** パラメーターを上書きします。

8 14 20 オプション: コントロールプレーンとコンピュートマシンの起動に使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS)。**platform.gcp.defaultMachinePlatform.osImage** の下の **project** および **name** パラメーターは、コントロールプレーンマシンとコンピュートマシンの両方に適用されます。**controlPlane.platform.gcp.osImage** または **compute.platform.gcp.osImage** の下の **project** および **name** パラメーターが設定されている場合、それらは **platform.gcp.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

16 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

21 既存 VPC の名前を指定します。

22 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

23 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

24 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

25 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

26 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

27 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

28 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

9.6.5.8. GCP にグローバルにアクセスできる Ingress コントローラーの作成

Google Cloud Platform (GCP) クラスターにグローバルにアクセスできる Ingress コントローラーを作成できます。グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

前提条件

- **install-config.yaml** を作成し、これに対する変更を完了している。

手順

グローバルアクセスが設定された Ingress コントローラーの新規の GCP クラスターへの作成

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストファイルを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-ingress-default-ingresscontroller.yaml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

出力例

```
cluster-ingress-default-ingresscontroller.yaml
```

3. エディターで **cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、必要な Operator 設定を記述するカスタムリソース (CR) を入力します。

サンプル clientAccess 設定を Global に設定します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1. **gcp.clientAccess** を **Global** に設定します。
2. グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

9.6.5.9. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

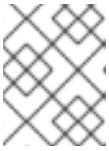
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.6.6. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

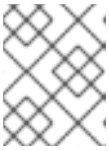
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.6.7. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように GCP クラスターを設定する の手順に従ってください。

9.6.7.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

例9.35 必要な GCP パーミッション

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete

- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
  ...
```

6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```

重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

9.6.7.2. 短期認証情報を使用するように GCP クラスターを設定

GCP Workload Identity を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な GCP リソースを作成する必要があります。

9.6.7.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- インストールプログラムが使用する GCP アカウントに次のいずれかの認証方法を追加している。
  - **IAM Workload Identity Pool Admin**ロール
  - 次の詳細な権限:

    例9.36 必要な GCP パーミッション

    - compute.projects.get
    - iam.googleapis.com/workloadIdentityPoolProviders.create
    - iam.googleapis.com/workloadIdentityPoolProviders.get
    - iam.googleapis.com/workloadIdentityPools.create
    - iam.googleapis.com/workloadIdentityPools.delete
    - iam.googleapis.com/workloadIdentityPools.get
    - iam.googleapis.com/workloadIdentityPools.undelete
    - iam.roles.create
    - iam.roles.delete
    - iam.roles.list
    - iam.roles.undelete

- iam.roles.update
- iam.serviceAccounts.create
- iam.serviceAccounts.delete
- iam.serviceAccounts.getIamPolicy
- iam.serviceAccounts.list
- iam.serviceAccounts.setIamPolicy
- iam.workloadIdentityPoolProviders.get
- iam.workloadIdentityPools.delete
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy
- storage.buckets.create
- storage.buckets.delete
- storage.buckets.get
- storage.buckets.getIamPolicy
- storage.buckets.setIamPolicy
- storage.objects.create
- storage.objects.delete
- storage.objects.list

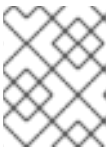
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

9.6.7.2.2. Cloud Credential Operator ユーティリティーを使用した GCP リソースの作成

ccoctl gcp create-all コマンドを使用して、GCP リソースの作成を自動化できます。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl gcp create-all \
  --name=<name> \ 1
  --region=<gcp_region> \ 2
  --project=<gcp_project_id> \ 3
  --credentials-requests-dir=<path_to_credentials_requests_directory> 4
```

1. トラッキングに使用される、作成されたすべての GCP リソースのユーザー定義名を指定します。
2. クラウドリソースを作成する GCP リージョンを指定します。
3. クラウドリソースを作成する GCP プロジェクト ID を指定します。
4. GCP サービスアカウントを作成するには、**CredentialsRequest** マニフェストのファイルが含まれるディレクトリーを指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

GCP にクエリーを実行すると、IAM サービスアカウントが作成されていることを確認できます。詳細については、IAM サービスアカウントのリスト表示に関する GCP のドキュメントを参照してください。

9.6.7.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

   例9.37 必要な GCP パーミッション

   - compute.machineTypes.list
   - compute.regions.list
   - compute.zones.list
   - dns.changes.create
   - dns.changes.get
   - dns.managedZones.create
   - dns.managedZones.delete
   - dns.managedZones.get
   - dns.managedZones.list
   - dns.networks.bindPrivateDNSZone
   - dns.resourceRecordSets.create
   - dns.resourceRecordSets.delete
   - dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

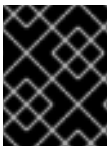
   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

9.6.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

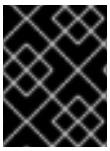
3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。

- **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

9.6.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

9.6.10. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

9.6.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを

追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.6.12. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。

9.7. GCP のクラスターの既存 VPC へのインストール

OpenShift Container Platform バージョン 4.16 では、Google Cloud Platform (GCP) 上の既存の Virtual Private Cloud (VPC) にクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

9.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

9.7.2. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、Google Cloud Platform (GCP) の既存の Virtual Private Cloud (VPC) 内にある既存サブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の GCP VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。サブネットのネットワークを設定する必要があります。

9.7.2.1. VPC を使用するための要件

VPC CIDR ブロックとマシンネットワーク CIDR の組み合わせは、空であってはなりません。サブネットはマシンネットワーク内にある必要があります。

インストールプログラムでは、次のコンポーネントは作成されません。

- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC ネットワーク

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

9.7.2.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- コントロールプレーンマシン用に 1 つのサブネットを提供し、コンピューティングマシン用に 1 つのサブネットを提供します。
- サブネットの CIDR は指定されたマシン CIDR に属します。

9.7.2.3. パーミッションの区分

一部の個人は、クラウド内に他のリソースとは異なるリソースを作成できます。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

9.7.2.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体で許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

9.7.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.7.4. クラスターノードの SSH アクセス用のキーペアの生成

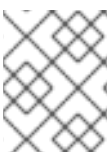
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

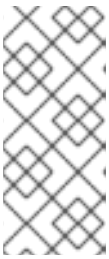
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

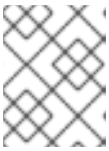
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

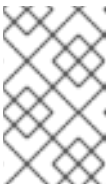
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定しま

9 。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.7.5. インストールプログラムの取得

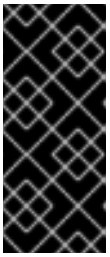
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

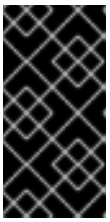
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

9.7.6. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、ファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

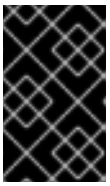
v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- GCP のインストール設定パラメーター

9.7.6.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

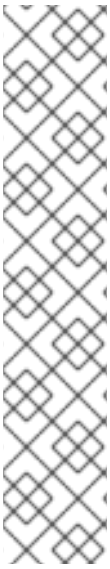
表9.19 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

9.7.6.2. GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform インスタンスタイプは OpenShift Container Platform でテストされています。

例9.38 マシンのシリーズ

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**
- **E2**

- **M1**
- **N1**
- **N2**
- **N2D**
- **Tau T2D**

9.7.6.3. 64 ビット ARM インフラストラクチャー上の GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform (GCP) 64 ビット ARM インスタンスタイプは OpenShift Container Platform でテストされています。

例9.39 64 ビット ARM マシン用のマシンシリーズ

- **Tau T2A**

9.7.6.4. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプは、コントロールプレーンとコンピューティングマシンの最小リソース要件を満たす必要があります。詳細については、「クラスターインストールの最小リソース要件」を参照してください。
- カスタムマシンタイプの名前は、次の構文に従う必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

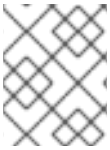
カスタムマシンタイプのサンプル install-config.yaml ファイル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
```

```
  gcp:
    type: custom-6-20480
replicas: 3
```

9.7.6.5. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。



注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

  b. コンピューティングマシンのみに Shielded VM を使用するには:

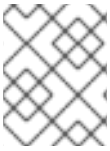
```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

  c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.7.6.6. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中のデータを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。



注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled 1
      type: n2d-standard-8 2
      onHostMaintenance: Terminate 3
```

1 Confidential VM を有効にします。

2 Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタイプの詳細については、サポートされているオペレーティングシステムとマシンタイプ を参照してください。

3 ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

  b. コンピューティングマシンのみに Confidential VM を使用するには:

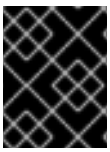
```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

  c. すべてのマシンに Confidential VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.7.6.7. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (6)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
      tags: (7)
      - control-plane-tag1
      - control-plane-tag2
      osImage: (8)
        project: example-project-name
        name: example-image-name
  replicas: 3
compute: (9) (10)
- hyperthreading: Enabled (11)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (12)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
```

```
          location: global
          projectID: project-id
        tags: 13
        - compute-tag1
        - compute-tag2
        osImage: 14
          project: example-project-name
          name: example-image-name
  replicas: 3
metadata:
  name: test-cluster 15
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 16
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 17
    region: us-central1 18
    defaultMachinePlatform:
      tags: 19
      - global-tag1
      - global-tag2
      osImage: 20
        project: example-project-name
        name: example-image-name
    network: existing_vpc 21
    controlPlaneSubnet: control_plane_subnet 22
    computeSubnet: compute_subnet 23
pullSecret: '{"auths": ...}' 24
fips: false 25
sshKey: ssh-ed25519 AAAA... 26
```

1 15 17 18 24 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 9 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 10 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 11 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を

Disabled に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

6 12 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントに適切な権限を付与する方法の詳細については、マシン管理 → コンピュートマシンセットの作成→ GCP でのコンピューティングマシンセットの作成を参照してください。

7 13 19 オプション: コントロールプレーンまたはコンピューティングマシンセットに適用するネットワークタグのセット。**platform.gcp.defaultMachinePlatform.tags** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**compute.platform.gcp.tags** パラメーターまたは **controlPlane.platform.gcp.tags** パラメーターが設定されている場合は、**platform.gcp.defaultMachinePlatform.tags** パラメーターを上書きします。

8 14 20 オプション: コントロールプレーンとコンピュートマシンの起動に使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS)。**platform.gcp.defaultMachinePlatform.osImage** の下の **project** および **name** パラメーターは、コントロールプレーンマシンとコンピュートマシンの両方に適用されます。**controlPlane.platform.gcp.osImage** または **compute.platform.gcp.osImage** の下の **project** および **name** パラメーターが設定されている場合、それらは **platform.gcp.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

16 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

21 既存 VPC の名前を指定します。

22 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

23 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

25 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

26 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

関連情報

- コンピュートマシンセットの顧客管理の暗号鍵の有効化

9.7.6.8. GCP にグローバルにアクセスできる Ingress コントローラーの作成

Google Cloud Platform (GCP) クラスターにグローバルにアクセスできる Ingress コントローラーを作成できます。グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

前提条件

- **install-config.yaml** を作成し、これに対する変更を完了している。

手順

グローバルアクセスが設定された Ingress コントローラーの新規の GCP クラスターへの作成

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストファイルを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-ingress-default-ingresscontroller.yaml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

出力例

```
cluster-ingress-default-ingresscontroller.yaml
```

3. エディターで **cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、必要な Operator 設定を記述するカスタムリソース (CR) を入力します。

サンプル clientAccess 設定を Global に設定します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1 **gcp.clientAccess** を **Global** に設定します。

2 グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

9.7.6.9. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

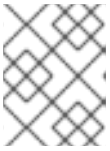
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

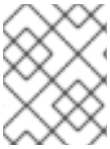
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

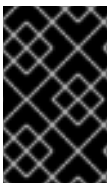


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.7.7. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.7.8. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように GCP クラスターを設定する の手順に従ってください。

9.7.8.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

例9.40 必要な GCP パーミッション

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete

- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
  ...
```

6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```

重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

9.7.8.2. 短期認証情報を使用するように GCP クラスターを設定

GCP Workload Identity を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な GCP リソースを作成する必要があります。

9.7.8.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- インストールプログラムが使用する GCP アカウントに次のいずれかの認証方法を追加している。
  - **IAM Workload Identity Pool Admin**ロール
  - 次の詳細な権限:

    例9.41 必要な GCP パーミッション

    - compute.projects.get
    - iam.googleapis.com/workloadIdentityPoolProviders.create
    - iam.googleapis.com/workloadIdentityPoolProviders.get
    - iam.googleapis.com/workloadIdentityPools.create
    - iam.googleapis.com/workloadIdentityPools.delete
    - iam.googleapis.com/workloadIdentityPools.get
    - iam.googleapis.com/workloadIdentityPools.undelete
    - iam.roles.create
    - iam.roles.delete
    - iam.roles.list
    - iam.roles.undelete

- iam.roles.update
- iam.serviceAccounts.create
- iam.serviceAccounts.delete
- iam.serviceAccounts.getIamPolicy
- iam.serviceAccounts.list
- iam.serviceAccounts.setIamPolicy
- iam.workloadIdentityPoolProviders.get
- iam.workloadIdentityPools.delete
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy
- storage.buckets.create
- storage.buckets.delete
- storage.buckets.get
- storage.buckets.getIamPolicy
- storage.buckets.setIamPolicy
- storage.objects.create
- storage.objects.delete
- storage.objects.list

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

9.7.8.2.2. Cloud Credential Operator ユーティリティーを使用した GCP リソースの作成

ccoctl gcp create-all コマンドを使用して、GCP リソースの作成を自動化できます。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

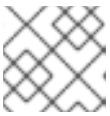
1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。



注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl gcp create-all \
  --name=<name> \ 1
  --region=<gcp_region> \ 2
  --project=<gcp_project_id> \ 3
  --credentials-requests-dir=<path_to_credentials_requests_directory> 4
```

1. トラッキングに使用される、作成されたすべての GCP リソースのユーザー定義名を指定します。
2. クラウドリソースを作成する GCP リージョンを指定します。
3. クラウドリソースを作成する GCP プロジェクト ID を指定します。
4. GCP サービスアカウントを作成するには、**CredentialsRequest** マニフェストのファイルが含まれるディレクトリーを指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

GCP にクエリーを実行すると、IAM サービスアカウントが作成されていることを確認できます。詳細については、IAM サービスアカウントのリスト表示に関する GCP のドキュメントを参照してください。

9.7.8.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

例9.42 必要な GCP パーミッション

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

9.7.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

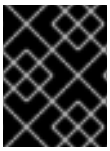
3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。

- **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

9.7.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

9.7.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.7.12. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

9.8. GCP 上のクラスターを共有 VPC にインストールする方法

OpenShift Container Platform バージョン 4.16 では、Google Cloud Platform (GCP) 上の共有 Virtual Private Cloud (VPC) にクラスターをインストールできます。このインストール方法では、クラスター

は別の GCP プロジェクトの VPC を使用するように設定されています。共有 VPC により、組織は複数のプロジェクトから共通の VPC ネットワークにリソースを接続できるようになります。対象のネットワークの内部 IP アドレスを使用して、組織内の通信を安全かつ効率的に実行できます。共有 VPC の詳細は、GCP ドキュメントの 共有 VPC の概要 を参照してください。

インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

9.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- 共有 VPC ネットワークを含む GCP ホストプロジェクトがある。
- クラスターをホストするように GCP プロジェクトを設定 している。サービスプロジェクトと呼ばれるこのプロジェクトは、ホストプロジェクトに割り当てる必要があります。詳細は、Attaching service projects in the GCP documentation を参照してください。
- ホストプロジェクトとサービスプロジェクトの両方で 必要な GCP 権限 を持つ GCP サービスアカウントを持っている。

9.8.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.8.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラ

ムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの ~/**.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

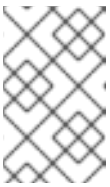
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.8.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

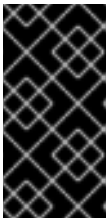
1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。

2. インフラストラクチャープロバイダーを選択します。

3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

9.8.5. GCP のインストール設定ファイルの作成

OpenShift Container Platform on Google Cloud Platform (GCP) を共有 VPC にインストールするには、**install-config.yaml** ファイルを生成し、クラスターが正しい VPC ネットワーク、DNS ゾーン、およびプロジェクト名を使用するように変更する必要があります。

9.8.5.1. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

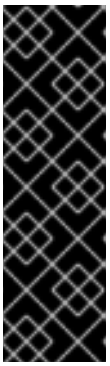
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

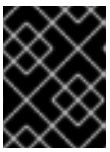
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- GCP のインストール設定パラメーター

9.8.5.2. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。

注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

b. コンピューティングマシンのみに Shielded VM を使用するには:

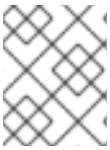
```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.8.5.3. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中のデータを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。



注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled (1)
      type: n2d-standard-8 (2)
      onHostMaintenance: Terminate (3)
```

(1) Confidential VM を有効にします。

2 Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタ

3 ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

b. コンピューティングマシンのみに Confidential VM を使用するには:

```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

c. すべてのマシンに Confidential VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.8.5.4. 共有 VPC インストール用にカスタマイズされた install-config.yaml ファイルのサンプル

共有 VPC を使用して OpenShift Container Platform を GCP にインストールするために必要な設定パラメーターがいくつかあります。以下は、これらのフィールドを示すサンプルの **install-config.yaml** ファイルです。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。このファイルを変更して、ご使用の環境とクラスターに適した値にする必要があります。

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Passthrough 1
metadata:
  name: cluster_name
platform:
  gcp:
    computeSubnet: shared-vpc-subnet-1 2
    controlPlaneSubnet: shared-vpc-subnet-2 3
    network: shared-vpc 4
    networkProjectID: host-project-name 5
    projectID: service-project-name 6
    region: us-east1
    defaultMachinePlatform:
```

```
      tags: 7
      - global-tag1
controlPlane:
  name: master
  platform:
    gcp:
      tags: 8
      - control-plane-tag1
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute:
- name: worker
  platform:
    gcp:
      tags: 9
      - compute-tag1
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA... 10
```

1 **credentialsMode** は **Passthrough** または **Manual** に設定する必要があります。サービスアカウントに必要な GCP 権限については、前提条件セクションを参照してください。

2 コンピュートマシンが使用する共有 VPC 内のサブネットの名前。

3 コントロールプレーンマシンが使用する共有 VPC 内のサブネットの名前。

4 共有 VPC の名前。

5 共有 VPC が存在するホストプロジェクトの名前。

6 クラスターをインストールする GCP プロジェクトの名前。

7 8 9 オプション:コンピューティングマシン、コントロールプレーンマシン、またはすべてのマシンに適用する 1 つ以上のネットワークタグ。

10 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

9.8.5.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

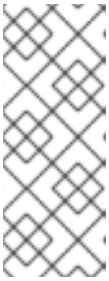
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを
5.

オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** で

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



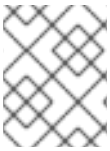
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

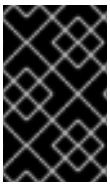


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.8.6. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.8.7. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように GCP クラスターを設定する の手順に従ってください。

9.8.7.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

例9.43 必要な GCP パーミッション

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get

- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。

3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
  ...
```

6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
```

```
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```

重要

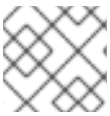
手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

9.8.7.2. 短期認証情報を使用するように GCP クラスターを設定

GCP Workload Identity を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な GCP リソースを作成する必要があります。

9.8.7.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- インストールプログラムが使用する GCP アカウントに次のいずれかの認証方法を追加している。
  - **IAM Workload Identity Pool Admin**ロール
  - 次の詳細な権限:

    例9.44 必要な GCP パーミッション

    - compute.projects.get
    - iam.googleapis.com/workloadIdentityPoolProviders.create
    - iam.googleapis.com/workloadIdentityPoolProviders.get
    - iam.googleapis.com/workloadIdentityPools.create
    - iam.googleapis.com/workloadIdentityPools.delete
    - iam.googleapis.com/workloadIdentityPools.get
    - iam.googleapis.com/workloadIdentityPools.undelete
    - iam.roles.create
    - iam.roles.delete

- iam.roles.list
- iam.roles.undelete
- iam.roles.update
- iam.serviceAccounts.create
- iam.serviceAccounts.delete
- iam.serviceAccounts.getIamPolicy
- iam.serviceAccounts.list
- iam.serviceAccounts.setIamPolicy
- iam.workloadIdentityPoolProviders.get
- iam.workloadIdentityPools.delete
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy
- storage.buckets.create
- storage.buckets.delete
- storage.buckets.get
- storage.buckets.getIamPolicy
- storage.buckets.setIamPolicy
- storage.objects.create
- storage.objects.delete
- storage.objects.list

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

9.8.7.2.2. Cloud Credential Operator ユーティリティーを使用した GCP リソースの作成

ccoctl gcp create-all コマンドを使用して、GCP リソースの作成を自動化できます。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

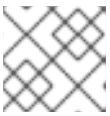
1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。



注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl gcp create-all \
  --name=<name> \ 1
  --region=<gcp_region> \ 2
  --project=<gcp_project_id> \ 3
  --credentials-requests-dir=<path_to_credentials_requests_directory> 4
```

1. トラッキングに使用される、作成されたすべての GCP リソースのユーザー定義名を指定します。
2. クラウドリソースを作成する GCP リージョンを指定します。
3. クラウドリソースを作成する GCP プロジェクト ID を指定します。
4. GCP サービスアカウントを作成するには、**CredentialsRequest** マニフェストのファイルが含まれるディレクトリーを指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

GCP にクエリーを実行すると、IAM サービスアカウントが作成されていることを確認できます。詳細については、IAM サービスアカウントのリスト表示に関する GCP のドキュメントを参照してください。

9.8.7.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

   例9.45 必要な GCP パーミッション

   - compute.machineTypes.list
   - compute.regions.list
   - compute.zones.list
   - dns.changes.create
   - dns.changes.get
   - dns.managedZones.create
   - dns.managedZones.delete
   - dns.managedZones.get
   - dns.managedZones.list
   - dns.networks.bindPrivateDNSZone
   - dns.resourceRecordSets.create
   - dns.resourceRecordSets.delete
   - dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

9.8.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

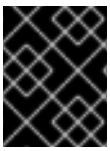
3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。

- **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

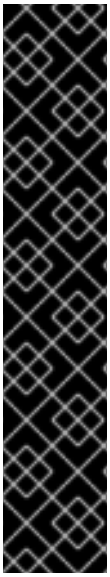


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

9.8.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

9.8.10. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.8.11. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

9.9. GCP へのプライベートクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、Google Cloud Platform (GCP) 上の既存の VPC にプライベートクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な

残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

9.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

9.9.2. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。
- 以下にアクセスできるマシンからデプロイ。
  - プロビジョニングするクラウドの API サービス。
  - プロビジョニングするネットワーク上のホスト。
  - インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

9.9.2.1. GCP のプライベートクラスター

Google Cloud Platform (GCP) にプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

クラスターには、GCP API にアクセスするためにインターネットへのアクセスが依然として必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックネットワークロードバランサー
- クラスターの **baseDomain** に一致するパブリック DNS ゾーン

インストールプログラムは、プライベート DNS ゾーンおよびクラスターに必要なレコードを作成するために指定する **baseDomain** を使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

ソースタグに基づいて外部ロードバランサーへのアクセスを制限できないため、プライベートクラスターは内部ロードバランサーのみを使用して内部インスタンスへのアクセスを許可します。

内部ロードバランサーは、ネットワークロードバランサーが使用するターゲットプールではなく、インスタンスグループに依存します。インストールプログラムは、グループにインスタンスがない場合でも、各ゾーンのインスタンスグループを作成します。

- クラスター IP アドレスは内部のみで使用されます。
- 1 つの転送ルールが Kubernetes API およびマシン設定サーバーポートの両方を管理します。
- バックエンドサービスは各ゾーンのインスタンスグループ、および存在する場合はブートストラップインスタンスグループで設定されます。
- ファイアウォールは、内部のソース範囲のみに基づく単一ルールを使用します。

9.9.2.1.1. 制限事項

ロードバランサーの機能の違いにより、マシン設定サーバー **/healthz** のヘルスチェックは実行されません。2 つの内部ロードバランサーが 1 つの IP アドレスを共有できませんが、2 つのネットワークロードバランサーは 1 つの外部 IP アドレスを共有できます。インスタンスが健全であるかどうかについては、ポート 6443 の **/readyz** チェックで完全に判別されます。

9.9.3. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、クラスターを Google Cloud Platform (GCP) の既存の VPC にデプロイできます。これを実行する場合、VPC 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の GCP VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

9.9.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなります。

- VPC

- サブネット
- Cloud Router
- Cloud NAT
- NAT IP アドレス

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、DHCP などの VPC オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

VPC およびサブネットは以下の要件を満たす必要があります。

- VPC は、OpenShift Container Platform クラスターをデプロイする同じ GCP プロジェクトに存在する必要があります。
- コントロールプレーンおよびコンピュートマシンからインターネットにアクセスできるようにするには、サブネットで Cloud NAT を設定してこれに対する egress を許可する必要があります。これらのマシンにパブリックアドレスがありません。インターネットへのアクセスが必要ない場合でも、インストールプログラムおよびイメージを取得できるように VPC ネットワークに対して egress を許可する必要があります。複数の Cloud NAT を共有サブネットで設定できないため、インストールプログラムはこれを設定できません。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定するすべてのサブネットが存在し、指定した VPC に属します。
- サブネットの CIDR はマシン CIDR に属します。
- クラスターのコントロールプレーンおよびコンピュートマシンをデプロイするためにサブネットを指定する必要があります。両方のマシンタイプに同じサブネットを使用できます。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。

9.9.3.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する GCP の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

9.9.3.3. クラスター間の分離

OpenShift Container Platform を既存ネットワークにデプロイする場合、クラスターサービスの分離は、クラスターのインフラストラクチャー ID によるクラスター内のマシンを参照するファイアウォールルールによって保持されます。クラスター内のトラフィックのみが許可されます。

複数のクラスターを同じ VPC にデプロイする場合、以下のコンポーネントはクラスター間のアクセスを共有する可能性があります。

- API: 外部公開ストラテジーでグローバルに利用可能か、内部公開ストラテジーのネットワーク全体で利用できる。
- デバッグツール: SSH および ICMP アクセス用にマシン CIDR に対して開かれている仮想マシンインスタンス上のポートなど。

9.9.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.9.5. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

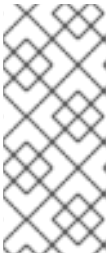
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

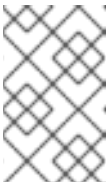
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

(1) **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.9.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

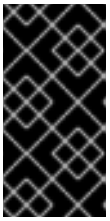
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

9.9.7. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- GCP のインストール設定パラメーター

9.9.7.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表9.20 最小リソース要件

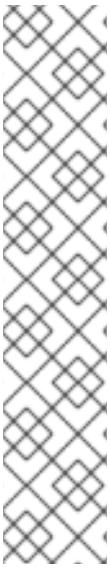
マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と

保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

9.9.7.2. GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform インスタンスタイプは OpenShift Container Platform でテストされています。

例9.46 マシンのシリーズ

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**
- **E2**
- **M1**
- **N1**
- **N2**
- **N2D**

- **Tau T2D**

9.9.7.3. 64 ビット ARM インフラストラクチャー上の GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform (GCP) 64 ビット ARM インスタンスタイプは OpenShift Container Platform でテストされています。

例9.47 64 ビット ARM マシン用のマシンシリーズ

- **Tau T2A**

9.9.7.4. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプは、コントロールプレーンとコンピューティングマシンの最小リソース要件を満たす必要があります。詳細については、「クラスターインストールの最小リソース要件」を参照してください。
- カスタムマシンタイプの名前は、次の構文に従う必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプのサンプル install-config.yaml ファイル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

9.9.7.5. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。

注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

  b. コンピューティングマシンのみに Shielded VM を使用するには:

```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

  c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.9.7.6. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中のデータを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。

注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled 1
      type: n2d-standard-8 2
      onHostMaintenance: Terminate 3
```

  1 Confidential VM を有効にします。

  2 Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタイプの詳細については、サポートされているオペレーティングシステムとマシンタイプ を参照してください。

  3 ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

  b. コンピューティングマシンのみに Confidential VM を使用するには:

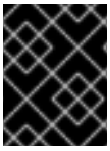
```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

  c. すべてのマシンに Confidential VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.9.7.7. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (6)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
      tags: (7)
      - control-plane-tag1
      - control-plane-tag2
      osImage: (8)
        project: example-project-name
        name: example-image-name
  replicas: 3
compute: (9) (10)
- hyperthreading: Enabled (11)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (12)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
        tags: (13)
        - compute-tag1
        - compute-tag2
```

```
      osImage: 14
        project: example-project-name
        name: example-image-name
  replicas: 3
metadata:
  name: test-cluster 15
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 16
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 17
    region: us-central1 18
    defaultMachinePlatform:
      tags: 19
      - global-tag1
      - global-tag2
      osImage: 20
        project: example-project-name
        name: example-image-name
    network: existing_vpc 21
    controlPlaneSubnet: control_plane_subnet 22
    computeSubnet: compute_subnet 23
pullSecret: '{"auths": ...}' 24
fips: false 25
sshKey: ssh-ed25519 AAAA... 26
publish: Internal 27
```

1 15 17 18 24 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: Cloud Credential Operator (CCO) に指定されたモードの使用を強制するには、このパラメーターを追加します。デフォルトでは、CCO は **kube-system** namespace のルート認証情報を使用して、認証情報の機能を動的に判断しようとします。CCO モードの詳細は、**認証および認可** ガイドの「Cloud Credential Operator について」セクションを参照してください。

3 9 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 10 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 11 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

6 12 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントに適切な権限を付与する方法の詳細については、マシン管理 → コンピュートマシンセットの作成→ GCP でのコンピューティングマシンセットの作成を参照してください。

7 13 19 オプション: コントロールプレーンまたはコンピューティングマシンセットに適用するネットワークタグのセット。**platform.gcp.defaultMachinePlatform.tags** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**compute.platform.gcp.tags** パラメーターまたは **controlPlane.platform.gcp.tags** パラメーターが設定されている場合は、**platform.gcp.defaultMachinePlatform.tags** パラメーターを上書きします。

8 14 20 オプション: コントロールプレーンとコンピュートマシンの起動に使用するカスタム Red Hat Enterprise Linux CoreOS (RHCOS)。**platform.gcp.defaultMachinePlatform.osImage** の下の **project** および **name** パラメーターは、コントロールプレーンマシンとコンピュートマシンの両方に適用されます。**controlPlane.platform.gcp.osImage** または **compute.platform.gcp.osImage** の下の **project** および **name** パラメーターが設定されている場合、それらは **platform.gcp.defaultMachinePlatform.osImage** パラメーターをオーバーライドします。

16 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

21 既存 VPC の名前を指定します。

22 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

23 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

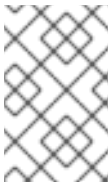
25 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

26 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

27 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

関連情報

- コンピュートマシンセットの顧客管理の暗号鍵の有効化

9.9.7.8. GCP にグローバルにアクセスできる Ingress コントローラーの作成

Google Cloud Platform (GCP) クラスターにグローバルにアクセスできる Ingress コントローラーを作成できます。グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

前提条件

- **install-config.yaml** を作成し、これに対する変更を完了している。

手順

グローバルアクセスが設定された Ingress コントローラーの新規の GCP クラスターへの作成

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストファイルを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-ingress-default-ingresscontroller.yaml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

出力例

```
cluster-ingress-default-ingresscontroller.yaml
```

3. エディターで **cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、必要な Operator 設定を記述するカスタムリソース (CR) を入力します。

サンプル clientAccess 設定を Global に設定します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1. **gcp.clientAccess** を **Global** に設定します。
2. グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

9.9.7.9. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

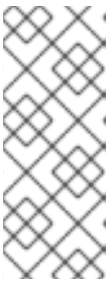
1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



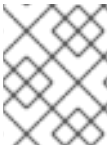
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.9.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.9.9. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

デフォルトでは、管理者のシークレットは **kube-system** プロジェクトに保存されます。**install-config.yaml** ファイルの **credentialsMode** パラメーターを **Manual** に設定した場合は、次のいずれかの代替手段を使用する必要があります。

- 長期クラウド認証情報を手動で管理するには、長期認証情報を手動で作成する の手順に従ってください。
- クラスターの外部で管理される短期認証情報を個々のコンポーネントに対して実装するには、短期認証情報を使用するように GCP クラスターを設定する の手順に従ってください。

9.9.9.1. 長期認証情報を手動で作成する

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

例9.48 必要な GCP パーミッション

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete

- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

   1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
   2. **install-config.yaml** ファイルの場所を指定します。
   3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
    ...
```

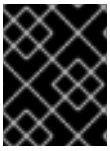
6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

シークレットを含む CredentialsRequest オブジェクトのサンプル

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
    ...
```

サンプル Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。

9.9.9.2. 短期認証情報を使用するように GCP クラスターを設定

GCP Workload Identity を使用するように設定されたクラスターをインストールするには、CCO ユーティリティーを設定し、クラスターに必要な GCP リソースを作成する必要があります。

9.9.9.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- インストールプログラムが使用する GCP アカウントに次のいずれかの認証方法を追加している。
  - **IAM Workload Identity Pool Admin**ロール
  - 次の詳細な権限:

    例9.49 必要な GCP パーミッション

    - compute.projects.get
    - iam.googleapis.com/workloadIdentityPoolProviders.create
    - iam.googleapis.com/workloadIdentityPoolProviders.get
    - iam.googleapis.com/workloadIdentityPools.create
    - iam.googleapis.com/workloadIdentityPools.delete
    - iam.googleapis.com/workloadIdentityPools.get
    - iam.googleapis.com/workloadIdentityPools.undelete
    - iam.roles.create
    - iam.roles.delete
    - iam.roles.list
    - iam.roles.undelete

- iam.roles.update
- iam.serviceAccounts.create
- iam.serviceAccounts.delete
- iam.serviceAccounts.getIamPolicy
- iam.serviceAccounts.list
- iam.serviceAccounts.setIamPolicy
- iam.workloadIdentityPoolProviders.get
- iam.workloadIdentityPools.delete
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy
- storage.buckets.create
- storage.buckets.delete
- storage.buckets.get
- storage.buckets.getIamPolicy
- storage.buckets.setIamPolicy
- storage.objects.create
- storage.objects.delete
- storage.objects.list

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

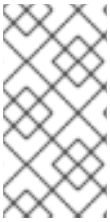
```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

9.9.9.2.2. Cloud Credential Operator ユーティリティーを使用した GCP リソースの作成

ccoctl gcp create-all コマンドを使用して、GCP リソースの作成を自動化できます。



注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

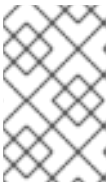
注記

このコマンドの実行には少し時間がかかる場合があります。

3. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl gcp create-all \
  --name=<name> \ 1
  --region=<gcp_region> \ 2
  --project=<gcp_project_id> \ 3
  --credentials-requests-dir=<path_to_credentials_requests_directory> 4
```

1. トラッキングに使用される、作成されたすべての GCP リソースのユーザー定義名を指定します。
2. クラウドリソースを作成する GCP リージョンを指定します。
3. クラウドリソースを作成する GCP プロジェクト ID を指定します。
4. GCP サービスアカウントを作成するには、**CredentialsRequest** マニフェストのファイルが含まれるディレクトリーを指定します。



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

GCP にクエリーを実行すると、IAM サービスアカウントが作成されていることを確認できます。詳細については、IAM サービスアカウントのリスト表示に関する GCP のドキュメントを参照してください。

9.9.9.2.3. Cloud Credential Operator ユーティリティーマニフェストの組み込み

個々のコンポーネントに対してクラスターの外部で管理される短期セキュリティー認証情報を実装するには、Cloud Credential Operator ユーティリティー (**ccoctl**) が作成したマニフェストファイルを、インストールプログラムの正しいディレクトリーに移動する必要があります。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- Cloud Credential Operator ユーティリティー (**ccoctl**) が設定されている。
- **ccoctl** ユーティリティーを使用して、クラスターに必要なクラウドプロバイダーリソースを作成している。

手順

1. インストールプログラムが使用する GCP アカウントに次の詳細な権限を追加します。

   例9.50 必要な GCP パーミッション

   - compute.machineTypes.list
   - compute.regions.list
   - compute.zones.list
   - dns.changes.create
   - dns.changes.get
   - dns.managedZones.create
   - dns.managedZones.delete
   - dns.managedZones.get
   - dns.managedZones.list
   - dns.networks.bindPrivateDNSZone
   - dns.resourceRecordSets.create
   - dns.resourceRecordSets.delete
   - dns.resourceRecordSets.list

2. **install-config.yaml** 設定ファイルの **credentialsMode** パラメーターを **Manual** に設定しなかった場合は、次のように値を変更します。

   設定ファイルのサンプルスニペット

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

3. インストールマニフェストファイルをまだ作成していない場合は、次のコマンドを実行して作成します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

4. 次のコマンドを実行して、**ccoctl** ユーティリティーが生成したマニフェストを、インストールプログラムが作成した **manifests** ディレクトリーにコピーします。

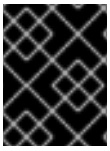
   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

5. 次のコマンドを実行して、**ccoctl** ユーティリティーが **tls** ディレクトリーに生成した秘密キーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

9.9.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

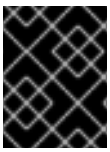
3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。

- **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

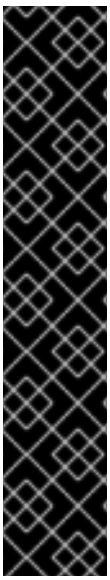


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

9.9.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

9.9.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.9.13. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

9.10. DEPLOYMENT MANAGER テンプレートの使用による GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、独自に提供するインフラストラクチャーを使用する Google Cloud Platform (GCP) にクラスターをインストールできます。

以下に、ユーザーによって提供されるインフラストラクチャーのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

9.10.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、長期間認証情報を手動で作成および維持 することができます。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

9.10.2. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

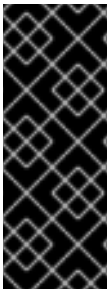
9.10.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.10.4. GCP プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

9.10.4.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。

重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

9.10.4.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。インストールに不要なオプションの API サービスを有効にすることもできます。GCP ドキュメントの サービスの有効化 を参照してください。

表9.21 必要な API サービス

API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**

表9.22 オプションの API サービス

API サービス	コンソールサービス名
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

9.10.4.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。

注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

9.10.4.4. GCP アカウントの制限

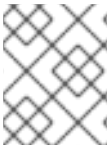
OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表9.23 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	6	1
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	Compute	グローバル	2	0
ヘルスチェック	Compute	グローバル	2	0
イメージ	Compute	グローバル	1	0
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
ルート	ネットワーク	グローバル	2	0
サブネットワーク	Compute	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0



注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

9.10.4.5. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. サービスアカウントキーを JSON 形式で作成するか、サービスアカウントを GCP 仮想マシンにアタッチできます。GCP ドキュメントの サービスアカウントキー の 作成とインスタンスのサービスアカウントの作成と有効 化をご覧ください。
クラスターを作成するには、サービスアカウントキーまたはサービスアカウントがアタッチされた仮想マシンが必要です。

注記

サービスアカウントがアタッチされた仮想マシンを使用してクラスターを作成する場合は、インストール前に **install-config.yaml** ファイルで **credentialsMode: Manual** を設定する必要があります。

9.10.4.6. 必要な GCP のロール

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。組織のセキュリティーポリシーでより制限的なアクセス許可のセットが必要な場合は、次のアクセス許可を持つサービスアカウントを作成できます。クラスターを既存の VPC (virtual private cloud) にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- ロール管理者
- Security Admin
- Service Account Admin
- サービスアカウントキー管理者
- サービスアカウントユーザー

- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS Administrator

Passthrough モードで Cloud Credential Operator を使用するために必要なロール

- ロードバランサー計算の管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor

次のロールは、コントロールプレーンとコンピュートマシンが使用するサービスアカウントに適用されます。

表9.24 GCP サービスアカウントのロール

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	**roles/compute.viewer**
	roles/storage.admin

9.10.4.7. ユーザーがプロビジョニングするインフラストラクチャーに必要な GCP 権限

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。

組織のセキュリティーポリシーで、より制限的なアクセス許可のセットが必要な場合は、必要なアクセス許可を持つ カスタムロール を作成できます。OpenShift Container Platform クラスターを作成および削除するには、ユーザーがプロビジョニングするインフラストラクチャーに以下のパーミッションが必要です。

> **例9.51 ネットワークリソースの作成に必要な権限**
>
> - **compute.addresses.create**
> - **compute.addresses.createInternal**

- compute.addresses.delete
- compute.addresses.get
- compute.addresses.list
- compute.addresses.use
- compute.addresses.useInternal
- compute.firewalls.create
- compute.firewalls.delete
- compute.firewalls.get
- compute.firewalls.list
- compute.forwardingRules.create
- compute.forwardingRules.get
- compute.forwardingRules.list
- compute.forwardingRules.setLabels
- compute.networks.create
- compute.networks.get
- compute.networks.list
- compute.networks.updatePolicy
- compute.routers.create
- compute.routers.get
- compute.routers.list
- compute.routers.update
- compute.routes.list
- compute.subnetworks.create
- compute.subnetworks.get
- compute.subnetworks.list
- compute.subnetworks.use
- compute.subnetworks.useExternalIp

例9.52 ロードバランサーリソースの作成に必要な権限

- compute.regionBackendServices.create

- compute.regionBackendServices.get
- compute.regionBackendServices.list
- compute.regionBackendServices.update
- compute.regionBackendServices.use
- compute.targetPools.addInstance
- compute.targetPools.create
- compute.targetPools.get
- compute.targetPools.list
- compute.targetPools.removeInstance
- compute.targetPools.use

例9.53 DNS リソースの作成に必要な権限

- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.list
- dns.resourceRecordSets.update

例9.54 サービスアカウントリソースの作成に必要な権限

- iam.serviceAccountKeys.create
- iam.serviceAccountKeys.delete
- iam.serviceAccountKeys.get
- iam.serviceAccountKeys.list
- iam.serviceAccounts.actAs
- iam.serviceAccounts.create
- iam.serviceAccounts.delete

- iam.serviceAccounts.get
- iam.serviceAccounts.list
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy

例9.55 コンピューティングリソースの作成に必要な権限

- compute.disks.create
- compute.disks.get
- compute.disks.list
- compute.instanceGroups.create
- compute.instanceGroups.delete
- compute.instanceGroups.get
- compute.instanceGroups.list
- compute.instanceGroups.update
- compute.instanceGroups.use
- compute.instances.create
- compute.instances.delete
- compute.instances.get
- compute.instances.list
- compute.instances.setLabels
- compute.instances.setMetadata
- compute.instances.setServiceAccount
- compute.instances.setTags
- compute.instances.use
- compute.machineTypes.get
- compute.machineTypes.list

例9.56 ストレージリソースの作成に必要

- storage.buckets.create

- **storage.buckets.delete**
- **storage.buckets.get**
- **storage.buckets.list**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.get**
- **storage.objects.list**

例9.57 ヘルスチェックリソースを作成するために必要な権限

- **compute.healthChecks.create**
- **compute.healthChecks.get**
- **compute.healthChecks.list**
- **compute.healthChecks.useReadOnly**
- **compute.httpHealthChecks.create**
- **compute.httpHealthChecks.get**
- **compute.httpHealthChecks.list**
- **compute.httpHealthChecks.useReadOnly**

例9.58 GCP ゾーンとリージョン関連の情報を取得するために必要な権限

- **compute.globalOperations.get**
- **compute.regionOperations.get**
- **compute.regions.list**
- **compute.zoneOperations.get**
- **compute.zones.get**
- **compute.zones.list**

例9.59 サービスとクォータを確認するために必要な権限

- **monitoring.timeSeries.list**
- **serviceusage.quotas.get**
- **serviceusage.services.list**

例9.60 インストールに必要な IAM パーミッション

- **iam.roles.get**

例9.61 インストールに必要なイメージ権限

- **compute.images.create**
- **compute.images.delete**
- **compute.images.get**
- **compute.images.list**

例9.62 収集ブートストラップを実行するためのオプションの権限

- **compute.instances.getSerialPortOutput**

例9.63 ネットワークリソースを削除するために必要な権限

- **compute.addresses.delete**
- **compute.addresses.deleteInternal**
- **compute.addresses.list**
- **compute.firewalls.delete**
- **compute.firewalls.list**
- **compute.forwardingRules.delete**
- **compute.forwardingRules.list**
- **compute.networks.delete**
- **compute.networks.list**
- **compute.networks.updatePolicy**
- **compute.routers.delete**
- **compute.routers.list**
- **compute.routes.list**
- **compute.subnetworks.delete**
- **compute.subnetworks.list**

例9.64 ロードバランサーリソースを削除するために必要な権限

- compute.regionBackendServices.delete
- compute.regionBackendServices.list
- compute.targetPools.delete
- compute.targetPools.list

例9.65 DNS リソースを削除するために必要な権限

- dns.changes.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

例9.66 サービスアカウントリソースを削除するために必要な権限

- iam.serviceAccounts.delete
- iam.serviceAccounts.get
- iam.serviceAccounts.list
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy

例9.67 コンピューティングリソースを削除するために必要な権限

- compute.disks.delete
- compute.disks.list
- compute.instanceGroups.delete
- compute.instanceGroups.list
- compute.instances.delete
- compute.instances.list
- compute.instances.stop
- compute.machineTypes.list

例9.68 ストレージリソースの削除に必要

- **storage.buckets.delete**
- **storage.buckets.getIamPolicy**
- **storage.buckets.list**
- **storage.objects.delete**
- **storage.objects.list**

例9.69 ヘルスチェックリソースを削除するために必要な権限

- **compute.healthChecks.delete**
- **compute.healthChecks.list**
- **compute.httpHealthChecks.delete**
- **compute.httpHealthChecks.list**

例9.70 削除に必要なイメージ権限

- **compute.images.delete**
- **compute.images.list**

例9.71 リージョン関連の情報を取得するために必要な権限

- **compute.regions.get**

例9.72 必要な Deployment Manager 権限

- **deploymentmanager.deployments.create**
- **deploymentmanager.deployments.delete**
- **deploymentmanager.deployments.get**
- **deploymentmanager.deployments.list**
- **deploymentmanager.manifests.get**
- **deploymentmanager.operations.get**
- **deploymentmanager.resources.list**

関連情報

- ストレージの最適化

9.10.4.8. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **africa-south1** (Johannesburg, South Africa)
- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-south2** (Delhi, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **australia-southeast2** (Melbourne, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-southwest1** (Madrid, Spain)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **europe-west8** (Milan, Italy)
- **europe-west9** (Paris, France)
- **europe-west12** (Turin, Italy)
- **me-central1** (Doha, Qatar, Middle East)
- **me-central2** (Dammam, Saudi Arabia, Middle East)

- **me-west1** (Tel Aviv, Israel)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **northamerica-northeast2** (Toronto, Ontario, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **southamerica-west1** (Santiago, Chile)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-east5** (Columbus, Ohio)
- **us-south1** (Dallas, Texas)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)

注記

リージョンおよびゾーンごとにどのマシンタイプのインスタンスが使用できるかを確認するには、Google の ドキュメント を参照してください。

9.10.4.9. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**
   - **gsutil**

   GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。
2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。

GCP ドキュメントで、サービスアカウントでの認証 を参照してください。

9.10.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

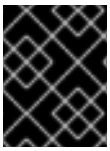
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

9.10.5.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表9.25 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

9.10.5.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表9.26 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

9.10.5.3. GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform インスタンスタイプは OpenShift Container Platform でテストされています。

例9.73 マシンのシリーズ

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**
- **E2**
- **M1**
- **N1**
- **N2**
- **N2D**
- **Tau T2D**

9.10.5.4. 64 ビット ARM インフラストラクチャー上の GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform (GCP) 64 ビット ARM インスタンスタイプは OpenShift Container Platform でテストされています。

例9.74 64 ビット ARM マシン用のマシンシリーズ

- **Tau T2A**

9.10.5.5. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプは、コントロールプレーンとコンピューティングマシンの最小リソース要件を満たす必要があります。詳細については、「クラスターインストールの最小リソース要件」を参照してください。
- カスタムマシンタイプの名前は、次の構文に従う必要があります。

custom-<number_of_cpus>-<amount_of_memory_in_mb>

たとえば、**custom-6-20480** です。

9.10.6. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

9.10.6.1. オプション: 別個の /var パーティションの作成

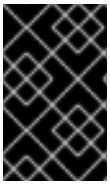
OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。



重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

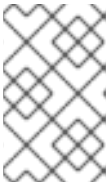
1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2

データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

9.10.6.2. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

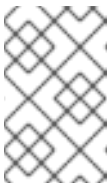
1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、ファイルへの絶対パスを入力する必要があります。

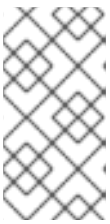
iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

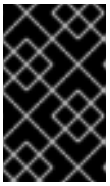
2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。



注記

3 ノードクラスターをインストールする場合は、必ず **compute.replicas** パラメーターを **0** に設定してください。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細については、「GCP に 3 ノードクラスターをインストールする」を参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

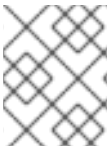
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- GCP のインストール設定パラメーター

9.10.6.3. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。



注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

  b. コンピューティングマシンのみに Shielded VM を使用するには:

```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

  c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.10.6.4. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中の

データを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。

注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled (1)
      type: n2d-standard-8 (2)
      onHostMaintenance: Terminate (3)
```

(1) Confidential VM を有効にします。

(2) Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタイプの詳細については、サポートされているオペレーティングシステムとマシンタイプ を参照してください。

(3) ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

  b. コンピューティングマシンのみに Confidential VM を使用するには:

```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

  c. すべてのマシンに Confidential VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
```

```
confidentialCompute: Enabled
type: n2d-standard-8
onHostMaintenance: Terminate
```

9.10.6.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3.

プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に .

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

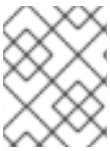
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



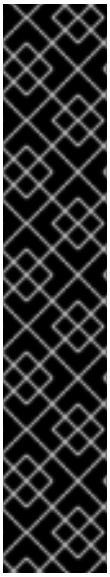
注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.10.6.6. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

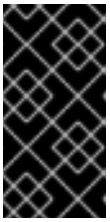
これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

4. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```



重要

ユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールするときに **MachineAPI** 機能を無効にした場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除する必要があります。そうしないと、クラスターのインストールに失敗します。

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- オプション: Ingress DNS レコードの追加

9.10.7. 一般的な変数のエクスポート

9.10.7.1. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な GCP リソースを見つけるためにも使用されます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

  -

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

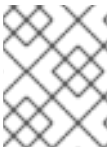
出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

9.10.7.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。



注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/17'
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

9.10.8. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/17**)。
4. **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.128.0/17**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

9.10.8.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例9.75 01_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
```

```
            }]
        }
    }]

    return {'resources': resources}
```

9.10.9. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

9.10.9.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

9.10.9.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表9.27 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。

プロトコル	ポート	説明
	10250-10259	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-9999	ポート **9100-9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000-32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表9.28 すべてのマシンからコントロールプレーンへの通信に使用されるポート

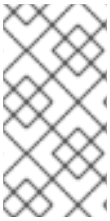
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表9.29 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

9.10.10. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブジェクトについて記述しています。
3. デプロイメントテンプレートが使用する変数をエクスポートします。
   a. クラスターネットワークの場所をエクスポートします。

   ```
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
   ```

   b. コントロールプレーンのサブネットの場所をエクスポートします。

   ```
   $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   c. クラスターが使用する 3 つのゾーンをエクスポートします。

   ```
   $ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
   ```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py 1
resources:
- name: cluster-lb-ext 2
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' 3
    region: '${REGION}' 4
```

```
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' 5
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: 6
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

1 2 外部クラスターをデプロイする場合にのみ必要です。

3 **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

4 **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

5 **control_subnet** は、コントロールサブセットの URL です。

6 **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

9.10.10.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

例9.76 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
```

```
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

9.10.10.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例9.77 02_lb_int.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-ig' + '.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
```

```
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-ig',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
```

```
    })

return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

9.10.11. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。
2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。

   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

   b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

9.10.11.1. プライベート DNS の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

例9.78 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

9.10.12. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。
2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。

(2) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(4) **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

9.10.12.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例9.79 03_firewall.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
```

```
            'ports': ['2379-2380']
        }],
        'sourceTags': [context.properties['infra_id'] + '-master'],
        'targetTags': [context.properties['infra_id'] + '-master']
    }
}, {
    'name': context.properties['infra_id'] + '-control-plane',
    'type': 'compute.v1.firewall',
    'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
            'IPProtocol': 'tcp',
            'ports': ['10257']
        },{
            'IPProtocol': 'tcp',
            'ports': ['10259']
        },{
            'IPProtocol': 'tcp',
            'ports': ['22623']
        }],
        'sourceTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [context.properties['infra_id'] + '-master']
    }
}, {
    'name': context.properties['infra_id'] + '-internal-network',
    'type': 'compute.v1.firewall',
    'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
            'IPProtocol': 'icmp'
        },{
            'IPProtocol': 'tcp',
            'ports': ['22']
        }],
        'sourceRanges': [context.properties['network_cidr']],
        'targetTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ]
    }
}, {
    'name': context.properties['infra_id'] + '-internal-cluster',
    'type': 'compute.v1.firewall',
    'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
            'IPProtocol': 'udp',
            'ports': ['4789', '6081']
        },{
            'IPProtocol': 'udp',
            'ports': ['500', '4500']
        },{
            'IPProtocol': 'esp',
```

```
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }]

    return {'resources': resources}
```

9.10.13. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。

2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コンピュートマシンをホストするサブネットの変数をエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
```

```
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

9.10.13.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例9.80 03_iam.py Deployment Manager テンプレート

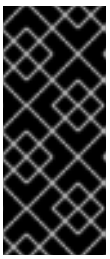
```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

9.10.14. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

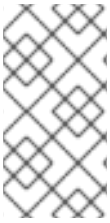
```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

9.10.15. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。

2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
```

```
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(3) **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。

(4) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(5) **control_subnet** は、コントロールサブセットの **selfLink** URL です。

(6) **image** は RHCOS イメージの **selfLink** URL です。

7 **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

8 **root_volume_size** はブートストラップマシンのブートディスクサイズです。

9 **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Deployment Manager の制限によりテンプレートではロードバランサーのメンバーシップを管理しないため、ブートストラップマシンは手動で追加する必要があります。

   a. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances \
    ${INFRA_ID}-bootstrap-ig --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

   b. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

```
$ gcloud compute backend-services add-backend \
    ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
group=${INFRA_ID}-bootstrap-ig --instance-group-zone=${ZONE_0}
```

9.10.15.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例9.81 04_bootstrap.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
```

```
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.2.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-ig',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

9.10.16. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    zones: (2)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' (3)
    image: '${CLUSTER_IMAGE}' (4)
    machine_type: 'n1-standard-4' (5)
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' (6)
```

```
  ignition: '${MASTER_IGNITION}' 7
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。
3. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
4. **image** は RHCOS イメージの **selfLink** URL です。
5. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
6. **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。
7. **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。
   - 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

     ```
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_0}-ig --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
     ```

     ```
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_1}-ig --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
     ```

     ```
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_2}-ig --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
     ```

   - 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

     ```
     $ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
     zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
     ```

     ```
     $ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
     zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
     ```

     ```
     $ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
     zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
     ```

9.10.16.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例9.82 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-1',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
```

```
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
```

```
    }]

    return {'resources': resources}
```

9.10.17. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-ig --instance-group-
zone=${ZONE_0}
```

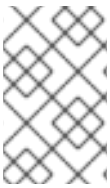
```
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
```

```
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
```

```
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

9.10.18. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。



注記

3 ノードクラスターをインストールする場合は、この手順をスキップしてください。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコントロールプレーンマシンで設定されます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。

注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
    a. コンピュートマシンをホストするサブネットをエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

b. サービスアカウントのメールアドレスをエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (9)
    zone: '${ZONE_1}' (10)
    compute_subnet: '${COMPUTE_SUBNET}' (11)
    image: '${CLUSTER_IMAGE}' (12)
    machine_type: 'n1-standard-4' (13)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (14)
    ignition: '${WORKER_IGNITION}' (15)
EOF
```

(1) **name** はワーカーマシンの名前です (例: **worker-0**)。

(2) (9) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) (10) **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。

(4) (11) **compute_subnet** はコンピュートサブネットの **selfLink** URL です。

(5) (12) **image** は RHCOS イメージの **selfLink** URL です。[1]

6 13 **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

7 14 **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。

8 15 **ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace イメージを使用するには、使用するオファーを指定します。

- OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736**
- OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
- OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**

9.10.18.1. ワーカーマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

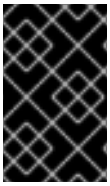
例9.83 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
```

```
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

9.10.19. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

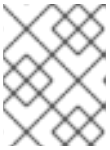
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。

5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.10.20. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

9.10.21. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

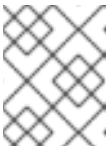
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

9.10.22. オプション: Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.*** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、 **EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP  98
```

2. A レコードをゾーンに追加します。

- A レコードを使用するには、以下を実行します。

i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

9.10.23. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. クラスターの稼働状態を確認します。

a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

```
$ oc get clusterversion
```

出力例

```
NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version            False       True          24m     Working towards 4.5.4: 99% complete
```

b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

```
$ oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                  NAME
READY   STATUS    RESTARTS   AGE
kube-system                                etcd-member-ip-10-0-3-111.us-east-
2.compute.internal             1/1     Running   0          35m
kube-system                                etcd-member-ip-10-0-3-239.us-east-
```

```
2.compute.internal              1/1       Running     0          37m
kube-system                                 etcd-member-ip-10-0-3-24.us-east-
2.compute.internal              1/1       Running     0          35m
openshift-apiserver-operator                openshift-apiserver-operator-6d6674f4f4-
h7t2t                   1/1       Running     1          37m
openshift-apiserver                         apiserver-fm48r
1/1       Running     0          30m
openshift-apiserver                         apiserver-fxkvv
1/1       Running     0          29m
openshift-apiserver                         apiserver-q85nm
1/1       Running     0          29m
...
openshift-service-ca-operator               openshift-service-ca-operator-66ff6dc6cd-
9r257                   1/1       Running     0          37m
openshift-service-ca                        apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1       Running     0          35m
openshift-service-ca                        configmap-cabundle-injector-8498544d7-
25qn6                   1/1       Running     0          35m
openshift-service-ca                        service-serving-cert-signer-6445fc9c6-wqdqn
1/1       Running     0          35m
openshift-service-catalog-apiserver-operator          openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w       1/1       Running     0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm     1/1       Running     0          31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

9.10.24. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.10.25. 次のステップ

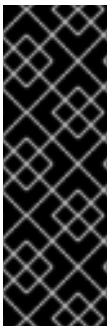
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- GCP にグローバルにアクセスできる Ingress コントローラーを設定 します。

9.11. DEPLOYMENT MANAGER テンプレートを使用した GCP の共有 VPC へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、独自に提供するインフラストラクチャーを使用する Google Cloud Platform (GCP) 上の共有 Virtual Private Cloud (VPC) に、クラスターをインストールできます。この場合、共有 VPC にインストールされたクラスターは、クラスターがデプロイされる場所とは異なるプロジェクトから VPC を使用するように設定されるクラスターです。

共有 VPC により、組織は複数のプロジェクトから共通の VPC ネットワークにリソースを接続できるようになります。対象のネットワークの内部 IP を使用して、組織内の通信を安全かつ効率的に実行できます。共有 VPC の詳細は、GCP ドキュメントの Shared VPC overview を参照してください。

以下に、ユーザーによって提供されるインフラストラクチャーの共有 VPC へのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

9.11.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、長期間認証情報を手動で作成および維持 することができます。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

9.11.2. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

9.11.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.11.4. クラスターをホストする GCP プロジェクトの設定

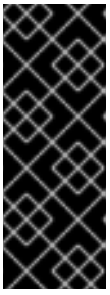
OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

9.11.4.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。



重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

9.11.4.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。インストールに不要なオプションの API サービスを有効にすることもできます。GCP ドキュメントの サービスの有効化 を参照してください。

表9.30 必要な API サービス

API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**

表9.31 オプションの API サービス

API サービス	コンソールサービス名
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

9.11.4.3. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表9.32 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	6	1
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	Compute	グローバル	2	0
ヘルスチェック	Compute	グローバル	2	0
イメージ	Compute	グローバル	1	0
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0
ルート	ネットワーク	グローバル	2	0
サブネットワーク	Compute	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0

注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**

- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

9.11.4.4. GCP でのサービスアカウントの作成

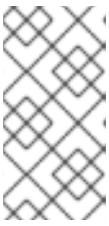
OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。



注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. サービスアカウントキーを JSON 形式で作成するか、サービスアカウントを GCP 仮想マシンにアタッチできます。GCP ドキュメントの サービスアカウントキー の 作成とインスタンスのサービスアカウントの作成と有効 化をご覧ください。
クラスターを作成するには、サービスアカウントキーまたはサービスアカウントがアタッチされた仮想マシンが必要です。

注記

サービスアカウントがアタッチされた仮想マシンを使用してクラスターを作成する場合は、インストール前に **install-config.yaml** ファイルで **credentialsMode: Manual** を設定する必要があります。

9.11.4.4.1. 必要な GCP のロール

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のイン

ストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。組織のセキュリティーポリシーでより制限的なアクセス許可のセットが必要な場合は、次のアクセス許可を持つサービスアカウントを作成できます。クラスターを既存の VPC (virtual private cloud) にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- ロール管理者
- Security Admin
- Service Account Admin
- サービスアカウントキー管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS Administrator

Passthrough モードで Cloud Credential Operator を使用するために必要なロール

- ロードバランサー計算の管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor

次のロールは、コントロールプレーンとコンピュートマシンが使用するサービスアカウントに適用されます。

表9.33 GCP サービスアカウントのロール

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	**roles/compute.viewer**

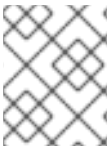
アカウント	ロール
	roles/storage.admin

9.11.4.5. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **africa-south1** (Johannesburg, South Africa)
- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-south2** (Delhi, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **australia-southeast2** (Melbourne, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-southwest1** (Madrid, Spain)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **europe-west8** (Milan, Italy)
- **europe-west9** (Paris, France)
- **europe-west12** (Turin, Italy)
- **me-central1** (Doha, Qatar, Middle East)

- **me-central2** (Dammam, Saudi Arabia, Middle East)
- **me-west1** (Tel Aviv, Israel)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **northamerica-northeast2** (Toronto, Ontario, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **southamerica-west1** (Santiago, Chile)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-east5** (Columbus, Ohio)
- **us-south1** (Dallas, Texas)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)



注記

リージョンおよびゾーンごとにどのマシンタイプのインスタンスが使用できるかを確認するには、Google の ドキュメント を参照してください。

9.11.4.6. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**
   - **gsutil**

   GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。

2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。
   GCP ドキュメントで、サービスアカウントでの認証 を参照してください。

9.11.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

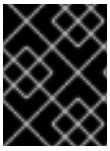
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

9.11.5.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表9.34 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

9.11.5.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

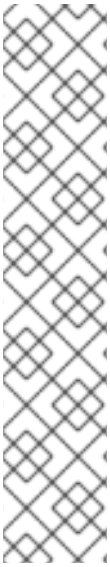
表9.35 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

9.11.5.3. GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform インスタンスタイプは OpenShift Container Platform でテストされています。

> **例9.84 マシンのシリーズ**
>
> - **A2**
> - **A3**
> - **C2**
> - **C2D**
> - **C3**
> - **C3D**
> - **E2**
> - **M1**
> - **N1**
> - **N2**
> - **N2D**
> - **Tau T2D**

9.11.5.4. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプは、コントロールプレーンとコンピューティングマシンの最小リソース要件を満たす必要があります。詳細については、「クラスターインストールの最小リソース要件」を参照してください。
- カスタムマシンタイプの名前は、次の構文に従う必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

9.11.6. 共有 VPC ネットワークをホストする GCP プロジェクトの設定

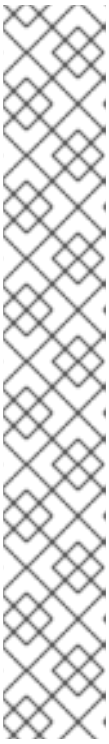
共有 VPC (Virtual Private Cloud) を使用して Google Cloud Platform (GCP) で OpenShift Container Platform クラスターをホストする場合、これをホストするプロジェクトを設定する必要があります。

注記

共有 VPC ネットワークをホストするプロジェクトがすでにある場合は、本セクションを参照して、プロジェクトが OpenShift Container Platform クラスターのインストールに必要なすべての要件を満たすことを確認します。

手順

1. OpenShift Container Platform クラスターの共有 VPC をホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。
2. 共有 VPC をホストするプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
3. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。



注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

共有 VPC ネットワークをホストするプロジェクトのサービスアカウントには以下のロールが必要です。

- コンピュートネットワークユーザー
- コンピュートセキュリティー管理者
- Deployment Manager Editor
- DNS 管理者
- セキュリティー管理者
- ネットワーク管理者

9.11.6.1. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、クラスターをインストールする共有 VPC をホストするプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。

注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

9.11.6.2. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

2. リソース定義で必要な以下の変数をエクスポートします。

   a. コントロールプレーンの CIDR をエクスポートします。

```
$ export MASTER_SUBNET_CIDR='10.0.0.0/17'
```

b. コンピュート CIDR をエクスポートします。

```
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'
```

c. VPC ネットワークおよびクラスターをデプロイするリージョンを以下にエクスポートします。

```
$ export REGION='<region>'
```

3. 共有 VPC をホストするプロジェクトの ID の変数をエクスポートします。

```
$ export HOST_PROJECT=<host_project>
```

4. ホストプロジェクトに属するサービスアカウントのメールの変数をエクスポートします。

```
$ export HOST_PROJECT_ACCOUNT=<host_service_account_email>
```

5. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '<prefix>' (1)
    region: '${REGION}' (2)
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' (3)
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' (4)
EOF
```

(1) **infra_id** は、ネットワーク名の接頭辞です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(3) **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/17**)。

(4) **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.128.0/17**)。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create <vpc_deployment_name> --config
01_vpc.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} (1)
```

(1) **<vpc_deployment_name>** には、デプロイする VPC の名前を指定します。

7. 他のコンポーネントが必要とする VPC 変数をエクスポートします。

a. ホストプロジェクトネットワークの名前をエクスポートします。

```
$ export HOST_PROJECT_NETWORK=<vpc_network>
```

b. ホストプロジェクトのコントロールプレーンのサブネットの名前をエクスポートします。

```
$ export HOST_PROJECT_CONTROL_SUBNET=<control_plane_subnet>
```

c. ホストプロジェクトのコンピュートサブネットの名前をエクスポートします。

```
$ export HOST_PROJECT_COMPUTE_SUBNET=<compute_subnet>
```

8. 共有 VPC を設定します。GCP ドキュメントの 共有 VPC の設定 を参照してください。

9.11.6.2.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例9.85 01_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
```

```
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }, {
            'name': context.properties['infra_id'] + '-nat-worker',
            'natIpAllocateOption': 'AUTO_ONLY',
            'minPortsPerVm': 512,
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }]
    }
}]

return {'resources': resources}
```

9.11.7. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

9.11.7.1. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

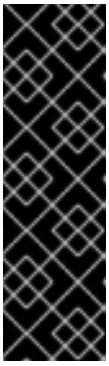
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

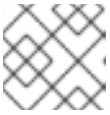
```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- GCP のインストール設定パラメーター

9.11.7.2. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。

注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
```

```
platform:
  gcp:
    secureBoot: Enabled
```

b. コンピューティングマシンのみに Shielded VM を使用するには:

```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.11.7.3. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中のデータを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。

注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled 1
      type: n2d-standard-8 2
      onHostMaintenance: Terminate 3
```

1 Confidential VM を有効にします。

2 Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタイプの詳細については、サポートされているオペレーティングシステムとマシンタイ

プ を参照してください。

3 ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

b. コンピューティングマシンのみに Confidential VM を使用するには:

```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

c. すべてのマシンに Confidential VM を使用するには:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.11.7.4. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      tags: 5
      - control-plane-tag1
      - control-plane-tag2
  replicas: 3
compute: 6
- hyperthreading: Enabled 7
```

```
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      tags: 8
      - compute-tag1
      - compute-tag2
  replicas: 0
metadata:
  name: test-cluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 9
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    defaultMachinePlatform:
      tags: 10
      - global-tag1
      - global-tag2
    projectID: openshift-production 11
    region: us-central1 12
pullSecret: '{"auths": ...}'
fips: false 13
sshKey: ssh-ed25519 AAAA... 14
publish: Internal 15
```

1 ホストプロジェクトでパブリック DNS を指定します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

5 8 10 オプション: コントロールプレーンまたはコンピューティングマシンセットに適用するネットワークタグのセット。**platform.gcp.defaultMachinePlatform.tags** パラメーターは、コントロールプレーンとコンピュートマシンの両方に適用されます。**compute.platform.gcp.tags** パラメーターまたは **controlPlane.platform.gcp.tags** パラメーターが設定されている場合は、**platform.gcp.defaultMachinePlatform.tags** パラメーターを上書きします。

9 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

11 仮想マシンインスタンスが存在するメインのプロジェクトを指定します。

12 VPC ネットワークが置かれているリージョンを指定します。

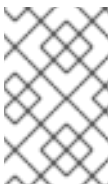
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

14 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

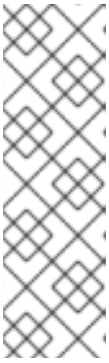
15 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。独自にプロビジョニングするインフラストラクチャーを使用するクラスターで共有 VPC を使用するには、**publish** を **Internal** に設定する必要があります。インストールプログラムは、ホストプロジェクトのベースドメインのパブリック DNS ゾーンにアクセスできなくなります。

9.11.7.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4.

指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

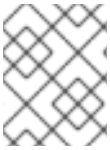
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.11.7.6. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

4. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

6. **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
status: {}
```

   1 このセクションを完全に削除します。

7. VPC のクラウドプロバイダーを設定します。

   a. **<installation_directory>/manifests/cloud-provider-config.yaml** ファイルを開きます。

   b. **network-project-id** パラメーターを追加し、その値を共有 VPC ネットワークをホストするプロジェクトの ID に設定します。

   c. **network-name** パラメーターを追加し、その値を OpenShift Container Platform クラスターをホストする共有 VPC ネットワークの名前に設定します。

   d. **subnetwork-name** パラメーターの値を、コンピュートマシンをホストする共有 VPC サブネットの値に置き換えます。

   <installation_directory>/manifests/cloud-provider-config.yaml の内容は以下の例のようになります。

```
config: |+
  [global]
  project-id     = example-project
  regional       = true
  multizone      = true
  node-tags      = opensh-ptzzx-master
  node-tags      = opensh-ptzzx-worker
  node-instance-prefix = opensh-ptzzx
  external-instance-groups-prefix = opensh-ptzzx
  network-project-id = example-shared-vpc
  network-name    = example-network
  subnetwork-name = example-worker-subnet
```

8. プライベートネットワーク上にないクラスターをデプロイする場合は、**<installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、**scope** パラメーターの値を **External** に置き換えます。ファイルの内容は以下の例のようになります。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
    type: LoadBalancerService
status:
  availableReplicas: 0
  domain: ''
  selector: ''
```

9. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

9.11.8. 一般的な変数のエクスポート

9.11.8.1. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な GCP リソースを見つけるためにも使用されます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

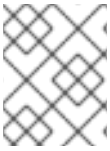
出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

9.11.8.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。



注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>' 1
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>' 2
```

```
$ export NETWORK_CIDR='10.0.0.0/16'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 3
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
```

1 2 ホストプロジェクトの値を指定します。

3 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

9.11.9. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

9.11.9.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

9.11.9.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表9.36 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック

プロトコル	ポート	説明
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表9.37 すべてのマシンからコントロールプレーンへの通信に使用されるポート

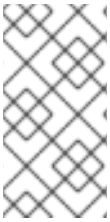
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表9.38 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

9.11.10. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブジェクトについて記述しています。
3. デプロイメントテンプレートが使用する変数をエクスポートします。
   a. クラスターネットワークの場所をエクスポートします。

   ```
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe
   ${HOST_PROJECT_NETWORK} --project ${HOST_PROJECT} --account
   ${HOST_PROJECT_ACCOUNT} --format json | jq -r .selfLink`)
   ```

   b. コントロールプレーンのサブネットの場所をエクスポートします。

   ```
   $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe
   ${HOST_PROJECT_CONTROL_SUBNET} --region=${REGION} --project
   ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
   .selfLink`)
   ```

   c. クラスターが使用する 3 つのゾーンをエクスポートします。

   ```
   $ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r
   .zones[0] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r
   .zones[1] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r
   .zones[2] | cut -d "/" -f9`)
   ```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 外部クラスターをデプロイする場合にのみ必要です。

(3) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(4) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(5) **control_subnet** は、コントロールサブセットの URL です。

(6) **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

9.11.10.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

例9.86 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

9.11.10.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例9.87 02_lb_int.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-ig' + '.selfLink)'
        })
```

```
    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-ig',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
```

```
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': zone
        }
    })

return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

9.11.11. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。
2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
```

```
  cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
  cluster_network: '${CLUSTER_NETWORK}' 3
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。
3. **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。

   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

   b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

9.11.11.1. プライベート DNS の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

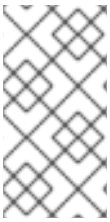
例9.88 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):
```

```
resources = [{
    'name': context.properties['infra_id'] + '-private-zone',
    'type': 'dns.v1.managedZone',
    'properties': {
        'description': '',
        'dnsName': context.properties['cluster_domain'] + '.',
        'visibility': 'private',
        'privateVisibilityConfig': {
            'networks': [{
                'networkUrl': context.properties['cluster_network']
            }]
        }
    }
}]

return {'resources': resources}
```

9.11.12. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。
2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
```

```
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。

(2) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(4) **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

9.11.12.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例9.89 03_firewall.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
```

```
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
```

```
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'udp',
                'ports': ['500', '4500']
            },{
                'IPProtocol': 'esp',
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }]

    return {'resources': resources}
```

9.11.13. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。
2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コントロールプレーンおよびコンピュートサブネットをホストするサブネットのサービスアカウントに、インストールプログラムが必要とするパーミッションを割り当てます。
   a. 共有 VPC をホストするプロジェクトの **networkViewer** ロールをマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
projects add-iam-policy-binding ${HOST_PROJECT} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role
"roles/compute.networkViewer"
```

b. **networkUser** ロールをコントロールプレーンサブネットのマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

c. **networkUser** ロールをコントロールプレーンサブネットのワーカーサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

d. **networkUser** ロールをコンピュートサブネットのマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

e. **networkUser** ロールをコンピュートサブネットのワーカーサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

9.11.13.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例9.90 03_iam.py Deployment Manager テンプレート

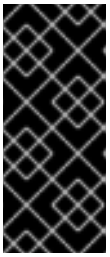
```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

9.11.14. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

9.11.15. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。

- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。

2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
```

```
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

3. **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。
4. **cluster_network** はクラスターネットワークの **selfLink** URL です。
5. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
6. **image** は RHCOS イメージの **selfLink** URL です。
7. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
8. **root_volume_size** はブートストラップマシンのブートディスクサイズです。
9. **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-bootstrap-ig --
zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

8. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

```
$ gcloud compute backend-services add-backend ${INFRA_ID}-api-internal-backend-service
--region=${REGION} --instance-group=${INFRA_ID}-bootstrap-ig --instance-group-
zone=${ZONE_0}
```

9.11.15.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

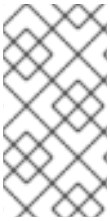
例9.9104_bootstrap.py Deployment Manager テンプレート

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-bootstrap-public-ip',
    'type': 'compute.v1.address',
    'properties': {
      'region': context.properties['region']
    }
  }, {
    'name': context.properties['infra_id'] + '-bootstrap',
    'type': 'compute.v1.instance',
    'properties': {
      'disks': [{
        'autoDelete': True,
        'boot': True,
```

```
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.2.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-ig',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

9.11.16. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    zones: (2)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' (3)
    image: '${CLUSTER_IMAGE}' (4)
    machine_type: 'n1-standard-4' (5)
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' (6)
```

```
  ignition: '${MASTER_IGNITION}' 7
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。
3. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
4. **image** は RHCOS イメージの **selfLink** URL です。
5. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
6. **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。
7. **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。
   - 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

     ```
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_0}-ig --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
     ```

     ```
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_1}-ig --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
     ```

     ```
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_2}-ig --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
     ```

   - 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

     ```
     $ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
     zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
     ```

     ```
     $ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
     zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
     ```

     ```
     $ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
     zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
     ```

9.11.16.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例9.92 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-1',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
```

```
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
```

```
    }]

    return {'resources': resources}
```

9.11.17. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-ig --instance-group-
zone=${ZONE_0}
```

```
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
```

```
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
```

```
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

9.11.18. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。

注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
   a. コンピュートマシンをホストするサブネットをエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_COMPUTE_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

   b. サービスアカウントのメールアドレスをエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (9)
    zone: '${ZONE_1}' (10)
    compute_subnet: '${COMPUTE_SUBNET}' (11)
    image: '${CLUSTER_IMAGE}' (12)
    machine_type: 'n1-standard-4' (13)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (14)
    ignition: '${WORKER_IGNITION}' (15)
EOF
```

(1) **name** はワーカーマシンの名前です (例: **worker-0**)。

(2) (9) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) (10) **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。

(4) (11) **compute_subnet** はコンピュートサブネットの **selfLink** URL です。

(5) (12) **image** は RHCOS イメージの **selfLink** URL です。[1]

(6) (13) **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

(7) (14) **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。

(8) (15) **Ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace イメージを使用するには、使用するオファーを指定します。

- OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736**
- OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
- OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**

9.11.18.1. ワーカーマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例9.93 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
```

```
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

9.11.19. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

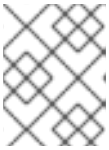
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

9.11.20. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

9.11.21. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

–

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

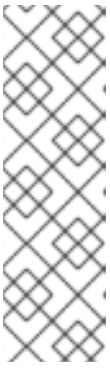
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

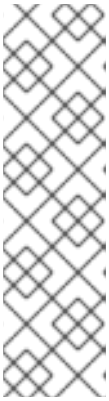
この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

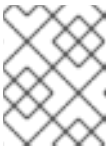
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

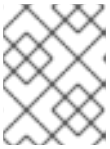
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

9.11.22. Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定が削除されます。Ingress ロードバランサーを参照する DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、 **EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP  98
```

2. A レコードをゾーンに追加します。

- A レコードを使用するには、以下を実行します。

i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME} -
-project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
```

- ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

9.11.23. Ingress ファイアウォールルールの追加

クラスターには複数のファイアウォールルールが必要です。共有 VPC を使用しない場合、これらのルールは GCP クラウドプロバイダーを介して Ingress コントローラーによって作成されます。共有 VPC を使用する場合は、現在すべてのサービスのクラスター全体のファイアウォールルールを作成するか、クラスターがアクセスを要求する際にイベントに基づいて各ルールを作成できます。クラスターがアクセスを要求する際に各ルールを作成すると、どのファイアウォールルールが必要であるかを正確に把握できます。クラスター全体のファイアウォールルールを作成する場合、同じルールセットを複数のクラスターに適用できます。

イベントに基づいて各ルールを作成する選択をする場合、クラスターをプロビジョニングした後、またはクラスターの有効期間中にコンソールがルールが見つからないことを通知する場合にファイアウォールルールを作成する必要があります。以下のイベントと同様のイベントが表示され、必要なファイアウォールルールを追加する必要があります。

```
$ oc get events -n openshift-ingress --field-selector="reason=LoadBalancerManualChange"
```

出力例

```
Firewall change required by security admin: `gcloud compute firewall-rules create k8s-fw-a26e631036a3f46cba28f8df67266d55 --network example-network --description "{\"kubernetes.io/service-name\":\"openshift-ingress/router-default\", \"kubernetes.io/service-ip\":\"35.237.236.234\"}\" --allow tcp:443,tcp:80 --source-ranges 0.0.0.0/0 --target-tags exampl-fqzq7-master,exampl-fqzq7-worker --project example-project`
```

これらのルールベースのイベントの作成時に問題が発生した場合には、クラスターの実行中にクラスター全体のファイアウォールルールを設定できます。

9.11.23.1. GCP での共有 VPC のクラスター全体のファイアウォールルールの作成

クラスター全体のファイアウォールルールを作成して、OpenShift Container Platform クラスターに必要なアクセスを許可します。

警告

クラスターイベントに基づいてファイアウォールルールを作成しない場合、クラスター全体のファイアウォールルールを作成する必要があります。

前提条件

- Deployment Manager テンプレートがクラスターをデプロイするために必要な変数をエクスポートしています。

- GCP にクラスターに必要なネットワークおよび負荷分散コンポーネントを作成しています。

手順

1. 単一のファイアウォールルールを追加して、Google Cloud Engine のヘルスチェックがすべてのサービスにアクセスできるようにします。このルールにより、Ingress ロードバランサーはインスタンスのヘルスステータスを判別できます。

```
$ gcloud compute firewall-rules create --allow='tcp:30000-32767,udp:30000-32767' --
network="${CLUSTER_NETWORK}" --source-
ranges='130.211.0.0/22,35.191.0.0/16,209.85.152.0/22,209.85.204.0/22' --target-
tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress-hc --
account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

2. 単一のファイアウォールルールを追加して、すべてのクラスターサービスへのアクセスを許可します。

   - 外部クラスターの場合:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
network="${CLUSTER_NETWORK}" --source-ranges="0.0.0.0/0" --target-
tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --
account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

   - プライベートクラスターの場合:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
network="${CLUSTER_NETWORK}" --source-ranges=${NETWORK_CIDR} --target-
tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --
account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

   このルールでは、TCP ポート **80** および **443** でのトラフィックのみを許可するため、サービスが使用するすべてのポートを追加するようにしてください。

9.11.24. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

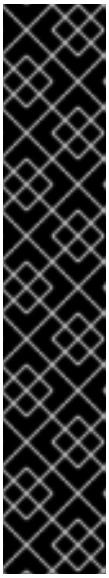
1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. クラスターの稼働状態を確認します。

   a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

   ```
   $ oc get clusterversion
   ```

   出力例

   ```
   NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
   version            False       True          24m     Working towards 4.5.4: 99% complete
   ```

   b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

   ```
   $ oc get clusteroperators
   ```

   出力例

   ```
   NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
   SINCE
   authentication                             4.5.4     True        False         False      7m56s
   cloud-credential                           4.5.4     True        False         False      31m
   cluster-autoscaler                         4.5.4     True        False         False      16m
   console                                    4.5.4     True        False         False      10m
   csi-snapshot-controller                    4.5.4     True        False         False      16m
   dns                                        4.5.4     True        False         False      22m
   etcd                                       4.5.4     False       False         False      25s
   image-registry                             4.5.4     True        False         False      16m
   ```

```
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                  NAME
READY   STATUS    RESTARTS   AGE
kube-system                                etcd-member-ip-10-0-3-111.us-east-
2.compute.internal             1/1       Running     0          35m
kube-system                                etcd-member-ip-10-0-3-239.us-east-
2.compute.internal             1/1       Running     0          37m
kube-system                                etcd-member-ip-10-0-3-24.us-east-
2.compute.internal             1/1       Running     0          35m
openshift-apiserver-operator               openshift-apiserver-operator-6d6674f4f4-
h7t2t                          1/1       Running     1          37m
openshift-apiserver                        apiserver-fm48r
1/1       Running     0          30m
openshift-apiserver                        apiserver-fxkvv
1/1       Running     0          29m
openshift-apiserver                        apiserver-q85nm
1/1       Running     0          29m
...
openshift-service-ca-operator              openshift-service-ca-operator-66ff6dc6cd-
9r257                          1/1       Running     0          37m
openshift-service-ca                       apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1       Running     0          35m
openshift-service-ca                       configmap-cabundle-injector-8498544d7-
25qn6                          1/1       Running     0          35m
openshift-service-ca                       service-serving-cert-signer-6445fc9c6-wqdqn
1/1       Running     0          35m
openshift-service-catalog-apiserver-operator          openshift-service-catalog-apiserver-
```

```
operator-549f44668b-b5q2w       1/1       Running     0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm     1/1       Running     0          31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

9.11.25. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.11.26. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

9.12. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、独自に提供するインフラストラクチャーとインストールリリースコンテンツの内部ミラーを使用する Google Cloud Platform (GCP) に、クラスターをインストールできます。

重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが GCP API を使用するにはインターネットへのアクセスが必要になります。

以下に、ユーザーによって提供されるインフラストラクチャーのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

9.12.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

  重要

  インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

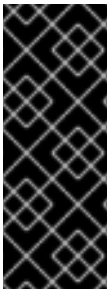
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。他のサイトへのアクセスを付与する必要がある場合もありますが、***.googleapis.com** および **accounts.google.com** へのアクセスを付与する必要があります。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、長期間認証情報を手動で作成および維持 することができます。

9.12.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

9.12.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

9.12.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

9.12.4. GCP プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

9.12.4.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。

重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

9.12.4.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。インストールに不要なオプションの API サービスを有効にすることもできます。GCP ドキュメントのサービスの有効化 を参照してください。

表9.39 必要な API サービス

API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**

表9.40 オプションの API サービス

API サービス	コンソールサービス名
Google Cloud API	**cloudapis.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**

API サービス	コンソールサービス名
Cloud Storage	**storage-component.googleapis.com**

9.12.4.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。

注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

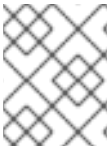
9.12.4.4. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表9.41 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	6	1
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	Compute	グローバル	2	0
ヘルスチェック	Compute	グローバル	2	0
イメージ	Compute	グローバル	1	0
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0
ルート	ネットワーク	グローバル	2	0
サブネットワーク	Compute	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0



注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**

- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

9.12.4.5. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. サービスアカウントキーを JSON 形式で作成するか、サービスアカウントを GCP 仮想マシンにアタッチできます。GCP ドキュメントの サービスアカウントキー の 作成とインスタンスのサービスアカウントの作成と有効 化をご覧ください。
クラスターを作成するには、サービスアカウントキーまたはサービスアカウントがアタッチされた仮想マシンが必要です。

注記

サービスアカウントがアタッチされた仮想マシンを使用してクラスターを作成する場合は、インストール前に **install-config.yaml** ファイルで **credentialsMode: Manual** を設定する必要があります。

9.12.4.6. 必要な GCP のロール

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。組織のセキュリティーポリシーでより制限的なアクセス許可のセットが必要な場合は、次のアクセス許可を持つサービスアカウントを作成できます。クラスターを既存の VPC (virtual private cloud) にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- ロール管理者
- Security Admin
- Service Account Admin
- サービスアカウントキー管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS Administrator

Passthrough モードで Cloud Credential Operator を使用するために必要なロール

- ロードバランサー計算の管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor

次のロールは、コントロールプレーンとコンピュートマシンが使用するサービスアカウントに適用されます。

表9.42 GCP サービスアカウントのロール

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	**roles/compute.viewer**

アカウント	ロール
	roles/storage.admin

9.12.4.7. ユーザーがプロビジョニングするインフラストラクチャーに必要な GCP 権限

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。

組織のセキュリティーポリシーで、より制限的なアクセス許可のセットが必要な場合は、必要なアクセス許可を持つ カスタムロール を作成できます。OpenShift Container Platform クラスターを作成および削除するには、ユーザーがプロビジョニングするインフラストラクチャーに以下のパーミッションが必要です。

例9.94 ネットワークリソースの作成に必要な権限

- **compute.addresses.create**
- **compute.addresses.createInternal**
- **compute.addresses.delete**
- **compute.addresses.get**
- **compute.addresses.list**
- **compute.addresses.use**
- **compute.addresses.useInternal**
- **compute.firewalls.create**
- **compute.firewalls.delete**
- **compute.firewalls.get**
- **compute.firewalls.list**
- **compute.forwardingRules.create**
- **compute.forwardingRules.get**
- **compute.forwardingRules.list**
- **compute.forwardingRules.setLabels**
- **compute.networks.create**
- **compute.networks.get**
- **compute.networks.list**
- **compute.networks.updatePolicy**

- compute.routers.create
- compute.routers.get
- compute.routers.list
- compute.routers.update
- compute.routes.list
- compute.subnetworks.create
- compute.subnetworks.get
- compute.subnetworks.list
- compute.subnetworks.use
- compute.subnetworks.useExternalIp

例9.95 ロードバランサーリソースの作成に必要な権限

- compute.regionBackendServices.create
- compute.regionBackendServices.get
- compute.regionBackendServices.list
- compute.regionBackendServices.update
- compute.regionBackendServices.use
- compute.targetPools.addInstance
- compute.targetPools.create
- compute.targetPools.get
- compute.targetPools.list
- compute.targetPools.removeInstance
- compute.targetPools.use

例9.96 DNS リソースの作成に必要な権限

- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.get
- dns.managedZones.list

- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.list
- dns.resourceRecordSets.update

例9.97 サービスアカウントリソースの作成に必要な権限

- iam.serviceAccountKeys.create
- iam.serviceAccountKeys.delete
- iam.serviceAccountKeys.get
- iam.serviceAccountKeys.list
- iam.serviceAccounts.actAs
- iam.serviceAccounts.create
- iam.serviceAccounts.delete
- iam.serviceAccounts.get
- iam.serviceAccounts.list
- resourcemanager.projects.get
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy

例9.98 コンピューティングリソースの作成に必要な権限

- compute.disks.create
- compute.disks.get
- compute.disks.list
- compute.instanceGroups.create
- compute.instanceGroups.delete
- compute.instanceGroups.get
- compute.instanceGroups.list
- compute.instanceGroups.update
- compute.instanceGroups.use
- compute.instances.create

- compute.instances.delete
- compute.instances.get
- compute.instances.list
- compute.instances.setLabels
- compute.instances.setMetadata
- compute.instances.setServiceAccount
- compute.instances.setTags
- compute.instances.use
- compute.machineTypes.get
- compute.machineTypes.list

例9.99 ストレージリソースの作成に必要

- storage.buckets.create
- storage.buckets.delete
- storage.buckets.get
- storage.buckets.list
- storage.objects.create
- storage.objects.delete
- storage.objects.get
- storage.objects.list

例9.100 ヘルスチェックリソースを作成するために必要な権限

- compute.healthChecks.create
- compute.healthChecks.get
- compute.healthChecks.list
- compute.healthChecks.useReadOnly
- compute.httpHealthChecks.create
- compute.httpHealthChecks.get
- compute.httpHealthChecks.list
- compute.httpHealthChecks.useReadOnly

例9.101 GCP ゾーンとリージョン関連の情報を取得するために必要な権限

- compute.globalOperations.get
- compute.regionOperations.get
- compute.regions.list
- compute.zoneOperations.get
- compute.zones.get
- compute.zones.list

例9.102 サービスとクォータを確認するために必要な権限

- monitoring.timeSeries.list
- serviceusage.quotas.get
- serviceusage.services.list

例9.103 インストールに必要な IAM パーミッション

- iam.roles.get

例9.104 インストールに必要なイメージ権限

- compute.images.create
- compute.images.delete
- compute.images.get
- compute.images.list

例9.105 収集ブートストラップを実行するためのオプションの権限

- compute.instances.getSerialPortOutput

例9.106 ネットワークリソースを削除するために必要な権限

- compute.addresses.delete
- compute.addresses.deleteInternal
- compute.addresses.list
- compute.firewalls.delete

- compute.firewalls.list
- compute.forwardingRules.delete
- compute.forwardingRules.list
- compute.networks.delete
- compute.networks.list
- compute.networks.updatePolicy
- compute.routers.delete
- compute.routers.list
- compute.routes.list
- compute.subnetworks.delete
- compute.subnetworks.list

例9.107 ロードバランサーリソースを削除するために必要な権限

- compute.regionBackendServices.delete
- compute.regionBackendServices.list
- compute.targetPools.delete
- compute.targetPools.list

例9.108 DNS リソースを削除するために必要な権限

- dns.changes.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

例9.109 サービスアカウントリソースを削除するために必要な権限

- iam.serviceAccounts.delete
- iam.serviceAccounts.get
- iam.serviceAccounts.list

- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**

例9.110 コンピューティングリソースを削除するために必要な権限

- **compute.disks.delete**
- **compute.disks.list**
- **compute.instanceGroups.delete**
- **compute.instanceGroups.list**
- **compute.instances.delete**
- **compute.instances.list**
- **compute.instances.stop**
- **compute.machineTypes.list**

例9.111 ストレージリソースの削除に必要

- **storage.buckets.delete**
- **storage.buckets.getIamPolicy**
- **storage.buckets.list**
- **storage.objects.delete**
- **storage.objects.list**

例9.112 ヘルスチェックリソースを削除するために必要な権限

- **compute.healthChecks.delete**
- **compute.healthChecks.list**
- **compute.httpHealthChecks.delete**
- **compute.httpHealthChecks.list**

例9.113 削除に必要なイメージ権限

- **compute.images.delete**
- **compute.images.list**

例9.114 リージョン関連の情報を取得するために必要な権限

- **compute.regions.get**

例9.115 必要な Deployment Manager 権限

- **deploymentmanager.deployments.create**
- **deploymentmanager.deployments.delete**
- **deploymentmanager.deployments.get**
- **deploymentmanager.deployments.list**
- **deploymentmanager.manifests.get**
- **deploymentmanager.operations.get**
- **deploymentmanager.resources.list**

関連情報

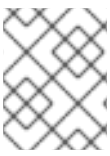
- ストレージの最適化

9.12.4.8. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **africa-south1** (Johannesburg, South Africa)
- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-south2** (Delhi, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **australia-southeast2** (Melbourne, Australia)
- **europe-central2** (Warsaw, Poland)

- **europe-north1** (Hamina, Finland)
- **europe-southwest1** (Madrid, Spain)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **europe-west8** (Milan, Italy)
- **europe-west9** (Paris, France)
- **europe-west12** (Turin, Italy)
- **me-central1** (Doha, Qatar, Middle East)
- **me-central2** (Dammam, Saudi Arabia, Middle East)
- **me-west1** (Tel Aviv, Israel)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **northamerica-northeast2** (Toronto, Ontario, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **southamerica-west1** (Santiago, Chile)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-east5** (Columbus, Ohio)
- **us-south1** (Dallas, Texas)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)



注記

リージョンおよびゾーンごとにどのマシンタイプのインスタンスが使用できるかを確認するには、Google の ドキュメント を参照してください。

9.12.4.9. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**
   - **gsutil**

   GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。
2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。
   GCP ドキュメントで、サービスアカウントでの認証 を参照してください。

9.12.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

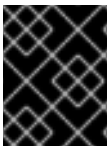
9.12.5.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表9.43 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。

ホスト	説明
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

9.12.5.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表9.44 最小リソース要件

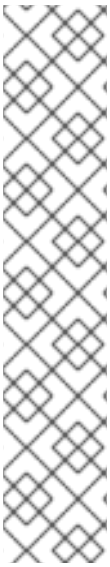
マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、そ

の他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

9.12.5.3. GCP のテスト済みインスタンスタイプ

以下の Google Cloud Platform インスタンスタイプは OpenShift Container Platform でテストされています。

例9.116 マシンのシリーズ

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**
- **E2**
- **M1**
- **N1**
- **N2**
- **N2D**
- **Tau T2D**

9.12.5.4. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプは、コントロールプレーンとコンピューティングマシンの最小リソース要件を満たす必要があります。詳細については、「クラスターインストールの最小リソース要件」を参照してください。
- カスタムマシンタイプの名前は、次の構文に従う必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

9.12.6. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

9.12.6.1. オプション: 別個の /var パーティションの作成

OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
```

```
- device: /dev/disk/by-partlabel/var
  path: /var
  format: xfs
  mount_options: [defaults, prjquota] 4
  with_mount_unit: true
```

1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

9.12.6.2. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。
      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、ファイルへの絶対パスを入力する必要があります。
      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。
      v. クラスターをデプロイするリージョンを選択します。
      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。
      vii. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

   a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

   ```
   pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
   "you@example.com"}}}'
   ```

   <mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

   b. **additionalTrustBundle** パラメーターおよび値を追加します。

   ```
   additionalTrustBundle: |
     -----BEGIN CERTIFICATE-----

   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
     -----END CERTIFICATE-----
   ```

   この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

   c. VPC のネットワークおよびサブネットを定義して、親の **platform.gcp** フィールドの下にクラスターをインストールします。

   ```
   network: <existing_vpc>
   controlPlaneSubnet: <control_plane_subnet>
   computeSubnet: <compute_subnet>
   ```

   platform.gcp.network には、既存の Google VPC の名前を指定します。**platform.gcp.controlPlaneSubnet** および **platform.gcp.computeSubnet** の場合には、コントロールプレーンマシンとコンピュートマシンをそれぞれデプロイするために既存のサブネットを指定します。

   d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

   ```
   imageContentSources:
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: quay.io/openshift-release-dev/ocp-release
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: registry.redhat.io/ocp/release
   ```

   これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

   e. オプション: パブリッシュストラテジーを **Internal** に設定します。

   ```
   publish: Internal
   ```

   このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

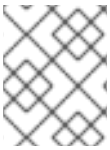
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- GCP のインストール設定パラメーター

9.12.6.3. Shielded VM の有効化

クラスターをインストールする場合は、Shielded VM を使用できます。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット検出などの追加のセキュリティー機能があります。詳細については、Shielded VM に関する Google のドキュメントを参照してください。



注記

Shielded VM は現在、64 ビット ARM インフラストラクチャーを備えたクラスターではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Shielded VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

  b. コンピューティングマシンのみに Shielded VM を使用するには:

```
compute:
- platform:
    gcp:
      secureBoot: Enabled
```

  c. すべてのマシンに Shielded VM を使用するには:

```
platform:
  gcp:
```

```
    defaultMachinePlatform:
      secureBoot: Enabled
```

9.12.6.4. Confidential VM の有効化

クラスターをインストールする場合は、Confidential VM を使用できます。Confidential VM は処理中のデータを暗号化します。詳細については、Confidential Computing に関する Google のドキュメントを参照してください。Confidential VM と Shielded VM を同時に有効にすることができますが、それらは互いに依存していません。

注記

現在、Confidential 仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

前提条件

- **install-config.yaml** ファイルを作成しました。

手順

- クラスターをデプロイする前に、テキストエディターを使用して、**install-config.yaml** ファイルを編集し、次のいずれかのスタンザを追加します。

  a. コントロールプレーンマシンのみに Confidential VM を使用するには:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled (1)
      type: n2d-standard-8 (2)
      onHostMaintenance: Terminate (3)
```

(1) Confidential VM を有効にします。

(2) Confidential VM をサポートするマシンタイプを指定します。Confidential VM には、N2D または C2D シリーズのマシンタイプが必要です。サポートされているマシンタイプの詳細については、サポートされているオペレーティングシステムとマシンタイプ を参照してください。

(3) ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

  b. コンピューティングマシンのみに Confidential VM を使用するには:

```
compute:
- platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

c. すべてのマシンに Confidential VM を使用するには:

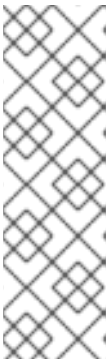
```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.12.6.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

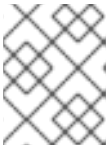
1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

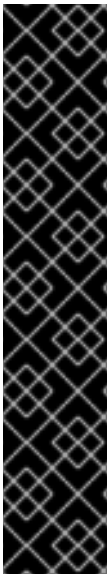
注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.12.6.6. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

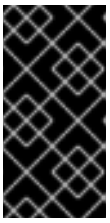
これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

4. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```



重要

ユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールするときに **MachineAPI** 機能を無効にした場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除する必要があります。そうしないと、クラスターのインストールに失敗します。

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- オプション: Ingress DNS レコードの追加

9.12.7. 一般的な変数のエクスポート

9.12.7.1. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な GCP リソースを見つけるためにも使用されます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

9.12.7.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。

注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

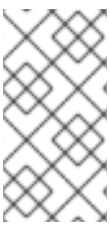
1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/17'
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

9.12.8. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/17**)。
4. **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.128.0/17**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

9.12.8.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例9.117 01_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
```

```
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

9.12.9. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

9.12.9.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

9.12.9.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表9.45 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

プロトコル	ポート	説明
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表9.46 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表9.47 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

9.12.10. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブジェクトについて記述しています。
3. デプロイメントテンプレートが使用する変数をエクスポートします。
    a. クラスターネットワークの場所をエクスポートします。

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
```

b. コントロールプレーンのサブネットの場所をエクスポートします。

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

c. クラスターが使用する 3 つのゾーンをエクスポートします。

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 外部クラスターをデプロイする場合にのみ必要です。

(3) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(4) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(5) **control_subnet** は、コントロールサブセットの URL です。

(6)

zones は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

9.12.10.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

例9.118 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
```

```
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

9.12.10.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例9.119 02_lb_int.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-ig' + '.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
```

```
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-ig',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

9.12.11. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。
2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。
   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

   b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

9.12.11.1. プライベート DNS の Deployment Manager テンプレート

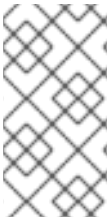
以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

例9.120 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

9.12.12. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。

2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。

(2) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(4) **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

9.12.12.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例9.121 03_firewall.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
```

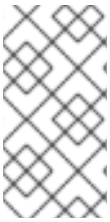
```
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
```

```
            'ports': ['22623']
        }],
        'sourceTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [context.properties['infra_id'] + '-master']
    }
}, {
    'name': context.properties['infra_id'] + '-internal-network',
    'type': 'compute.v1.firewall',
    'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
            'IPProtocol': 'icmp'
        },{
            'IPProtocol': 'tcp',
            'ports': ['22']
        }],
        'sourceRanges': [context.properties['network_cidr']],
        'targetTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ]
    }
}, {
    'name': context.properties['infra_id'] + '-internal-cluster',
    'type': 'compute.v1.firewall',
    'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
            'IPProtocol': 'udp',
            'ports': ['4789', '6081']
        },{
            'IPProtocol': 'udp',
            'ports': ['500', '4500']
        },{
            'IPProtocol': 'esp',
        },{
            'IPProtocol': 'tcp',
            'ports': ['9000-9999']
        },{
            'IPProtocol': 'udp',
            'ports': ['9000-9999']
        },{
            'IPProtocol': 'tcp',
            'ports': ['10250']
        },{
            'IPProtocol': 'tcp',
            'ports': ['30000-32767']
        },{
            'IPProtocol': 'udp',
            'ports': ['30000-32767']
        }],
        'sourceTags': [
            context.properties['infra_id'] + '-master',
```

```
            context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ]
    }
}]

return {'resources': resources}
```

9.12.13. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。
2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コンピュートマシンをホストするサブネットの変数をエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

9.12.13.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例9.122 03_iam.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
```

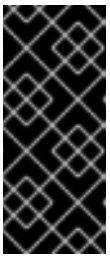
```
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

9.12.14. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

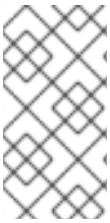
```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

9.12.15. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
```

```
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(3) **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。

(4) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(5) **control_subnet** は、コントロールサブセットの **selfLink** URL です。

(6) **image** は RHCOS イメージの **selfLink** URL です。

(7) **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

(8) **root_volume_size** はブートストラップマシンのブートディスクサイズです。

(9) **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Deployment Manager の制限によりテンプレートではロードバランサーのメンバーシップを管理しないため、ブートストラップマシンは手動で追加する必要があります。

   a. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances \
    ${INFRA_ID}-bootstrap-ig --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

b. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

```
$ gcloud compute backend-services add-backend \
    ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
group=${INFRA_ID}-bootstrap-ig --instance-group-zone=${ZONE_0}
```

9.12.15.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例9.123 04_bootstrap.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.2.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
```

```
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-ig',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

9.12.16. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述し

ています。

2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。
3. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
4. **image** は RHCOS イメージの **selfLink** URL です。
5. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
6. **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。
7. **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。

- 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-${ZONE_0}-ig --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
```

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-${ZONE_1}-ig --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
```

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-${ZONE_2}-ig --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
```

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
```

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

9.12.16.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例9.124 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
```

```
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][0]
    }
}, {
    'name': context.properties['infra_id'] + '-master-1',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][1]
    }
}, {
    'name': context.properties['infra_id'] + '-master-2',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
```

```
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
    }]

    return {'resources': resources}
```

9.12.17. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-ig --instance-group-zone=${ZONE_0}
```

```
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
```

```
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
```

```
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

9.12.18. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。

注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。

- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
   a. コンピュートマシンをホストするサブネットをエクスポートします。

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   b. サービスアカウントのメールアドレスをエクスポートします。

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

   c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

   ```
   $ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
   ```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
```

```
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
    compute_subnet: '${COMPUTE_SUBNET}' 11
    image: '${CLUSTER_IMAGE}' 12
    machine_type: 'n1-standard-4' 13
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
    ignition: '${WORKER_IGNITION}' 15
EOF
```

1 **name** はワーカーマシンの名前です (例: **worker-0**)。

2 9 **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3 10 **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。

4 11 **compute_subnet** はコンピュートサブネットの **selfLink** URL です。

5 12 **image** は RHCOS イメージの **selfLink** URL です。[1]

6 13 **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

7 14 **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。

8 15 **ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace イメージを使用するには、使用するオファーを指定します。

   - OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736**
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
   - OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**

9.12.18.1. ワーカーマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例9.125 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

9.12.19. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

9.12.20. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

9.12.21. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

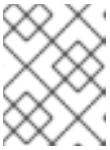
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

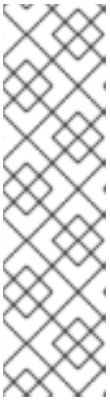
```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

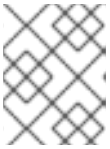
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

9.12.22. オプション: Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.*** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、 **EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP  98
```

2. A レコードをゾーンに追加します。

- A レコードを使用するには、以下を実行します。

i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

9.12.23. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. クラスターの稼働状態を確認します。
   a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

```
$ oc get clusterversion
```

出力例

```
NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version            False       True          24m     Working towards 4.5.4: 99% complete
```

b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

```
$ oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                  NAME
READY   STATUS    RESTARTS   AGE
kube-system                                etcd-member-ip-10-0-3-111.us-east-
2.compute.internal             1/1     Running   0          35m
kube-system                                etcd-member-ip-10-0-3-239.us-east-
```

```
2.compute.internal                1/1     Running     0          37m
kube-system                                 etcd-member-ip-10-0-3-24.us-east-
2.compute.internal                1/1     Running     0          35m
openshift-apiserver-operator                openshift-apiserver-operator-6d6674f4f4-
h7t2t                     1/1     Running     1          37m
openshift-apiserver                         apiserver-fm48r
1/1     Running     0          30m
openshift-apiserver                         apiserver-fxkvv
1/1     Running     0          29m
openshift-apiserver                         apiserver-q85nm
1/1     Running     0          29m
...
openshift-service-ca-operator               openshift-service-ca-operator-66ff6dc6cd-
9r257                     1/1     Running     0          37m
openshift-service-ca                        apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1     Running     0          35m
openshift-service-ca                        configmap-cabundle-injector-8498544d7-
25qn6                     1/1     Running     0          35m
openshift-service-ca                        service-serving-cert-signer-6445fc9c6-wqdqn
1/1     Running     0          35m
openshift-service-catalog-apiserver-operator          openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w       1/1     Running     0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm     1/1     Running     0          31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

9.12.24. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

9.12.25. 次のステップ

- クラスターをカスタマイズ します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。

- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。

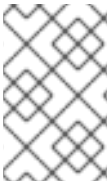
9.13. GCP に 3 ノードクラスターをインストールする

OpenShift Container Platform バージョン 4.16 では、Google Cloud Platform (GCP) に 3 ノードクラスターをインストールできます。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコントロールプレーンマシンで設定されます。このタイプのクラスターは、クラスター管理者および開発者がテスト、開発、および実稼働に使用するためのより小さくリソース効率の高いクラスターを提供します。

インストーラーによってプロビジョニングされたインフラストラクチャーまたはユーザーによってプロビジョニングされたインフラストラクチャーのいずれかを使用して、3 ノードクラスターをインストールできます。

9.13.1. 3 ノードクラスターの設定

クラスターをデプロイする前に、**install-config.yaml** ファイルでワーカーノードの数を **0** に設定して、3 ノードクラスターを設定します。ワーカーノードの数を **0** に設定すると、コントロールプレーンマシンがスケジュール可能になります。これにより、アプリケーションワークロードをコントロールプレーンノードから実行するようにスケジュールできます。



注記

アプリケーションワークロードはコントロールプレーンノードから実行され、コントロールプレーンノードはコンピュートノードと見なされるため、追加のサブスクリプションが必要です。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

1. 次の **compute** スタンザに示すように、**install-config.yaml** ファイルでコンピューティングレプリカの数を **0** に設定します。

   3 ノードクラスターの install-config.yaml ファイルの例

   ```
   apiVersion: v1
   baseDomain: example.com
   compute:
   - name: worker
     platform: {}
     replicas: 0
   # ...
   ```

2. ユーザーがプロビジョニングしたインフラストラクチャーを使用して、クラスターをデプロイする場合:

   - Kubernetes マニフェストファイルを作成したら、**cluster-scheduler-02-config.yml** ファイルで **spec.mastersSchedulable** パラメーターが **true** に設定されていることを確認します。このファイルは、**<installation_directory>/manifests** にあります。詳細については、

「Deployment Manager テンプレートを使用して、GCP でユーザーがプロビジョニングしたインフラストラクチャーにクラスターをインストールする」の「Kubernetes マニフェストと Ignition 設定ファイルの作成」を参照してください。

- 追加のワーカーノードを作成しないでください。

3 ノードクラスターの cluster-scheduler-02-config.yml ファイルの例

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  creationTimestamp: null
  name: cluster
spec:
  mastersSchedulable: true
  policy:
    name: ""
status: {}
```

9.13.2. 次のステップ

- カスタマイズによる GCP へのクラスターのインストール
- Deployment Manager テンプレートの使用による GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

9.14. GCP のインストール設定パラメーター

OpenShift Container Platform クラスターを Google Cloud Platform (GCP) にデプロイする前に、パラメーターを指定してクラスターとそれをホストするプラットフォームをカスタマイズします。**install-config.yaml** ファイルを作成するときは、コマンドラインを使用して必要なパラメーターの値を指定します。その後、**install-config.yaml** ファイルを変更して、クラスターをさらにカスタマイズできます。

9.14.1. GCP で使用可能なインストール設定パラメーター

次の表に、インストールプロセスの一部として設定できる必須、オプション、および GCP 固有のインストール設定パラメーターを示します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

9.14.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表9.48 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object

パラメーター	説明	値
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

9.14.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表9.49 ネットワークパラメーター

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

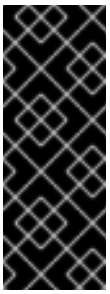
パラメーター	説明	値
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

9.14.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表9.50 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列

パラメーター	説明	値
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** と **arm64** です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**

パラメーター	説明	値
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** と **arm64** です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。 **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**

パラメーター	説明	値
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。	**false** または **true**

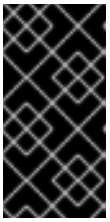
パラメーター	説明	値
	重要  クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String

パラメーター	説明	値
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

注記

GCP で共有 Virtual Private Cloud (VPC) にインストールする場合は、**credentialsMode** を **Passthrough** または **Manual** に設定する必要があります。



重要

このパラメーターを **Manual** に設定すると、管理者レベルのシークレットを **kube-system** プロジェクトに保存する代替手段が有効になりますが、追加の設定手順が必要になります。詳細は、管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法を参照してください。

9.14.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

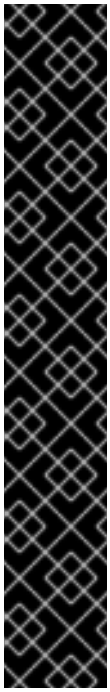
表9.51 追加の GCP パラメーター

パラメーター	説明	値
controlPlane: platform: gcp: osImage: project:	オプション: デフォルトで、インストールプログラムは、コントロールプレーンおよびコンピュートマシンの起動に使用する Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードしてインストールします。インストールプログラムがコントロールプレーンマシンのみに使用するカスタム RHCOS イメージの場所を指定することで、デフォルトの動作をオーバーライドできます。	文字列。イメージが置かれている GCP プロジェクトの名前。

パラメーター	説明	値
controlPlane: platform: gcp: osImage: name:	インストールプログラムがコントロールプレーンマシンを起動するために使用するカスタム RHCOS イメージの名前。**controlPlane.platform.gcp.osImage.project** を使用する場合、このフィールドは必須です。	文字列。RHCOS イメージの名前。
compute: platform: gcp: osImage: project:	オプション: デフォルトで、インストールプログラムはコンピュートマシンの起動に使用する RHCOS イメージをダウンロードしてインストールします。インストールプログラムがコンピュートマシンのみに使用するカスタム RHCOS イメージの場所を指定することで、デフォルトの動作をオーバーライドできます。	文字列。イメージが置かれている GCP プロジェクトの名前。

パラメーター	説明	値
compute: platform: gcp: osImage: name:	インストールプログラムがコンピュートマシンの起動に使用するカスタム RHCOS イメージの名前。**compute.platform.gcp.osImage.project** を使用する場合、このフィールドは必須です。	文字列。RHCOS イメージの名前。
platform: gcp: network:	クラスターをデプロイする既存 Virtual Private Cloud (VPC) の名前。クラスターを共有 VPC にデプロイする場合は、共有 VPC を含む GCP プロジェクトの名前で **platform.gcp.networkProjectID** を設定する必要があります。	文字列。
platform: gcp: networkProjectID:	オプション: クラスターをデプロイする共有 VPC を含む GCP プロジェクトの名前。	文字列。

パラメーター	説明	値
platform: gcp: projectID:	インストールプログラムがクラスターをインストールする GCP プロジェクトの名前。	文字列。
platform: gcp: region:	クラスターをホストする GCP リージョンの名前。	有効なリージョン名 (例: **us-central1)**。
platform: gcp: controlPlaneSubnet:	コントロールプレーンマシンをデプロイする既存サブネットの名前。	サブネット名。

パラメーター	説明	値
platform: gcp: computeSubnet:	コンピュートマシンをデプロイする既存サブネットの名前。	サブネット名。
platform: gcp: defaultMachinePlatform: zones:	インストールプログラムがマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。  **重要** GCP 64 ビット ARM インフラストラクチャーでクラスターを実行する場合は、Ampere Altra Arm CPU が利用可能なゾーンを使用するようにしてください。「GCP 可用性ゾーン」リンクで、64 ビット ARM プロセッサーと互換性のあるゾーンを確認できます。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: osDisk: diskSizeGB:	ディスクのサイズ (GB 単位)。	16 GB から 65536 GB の間のサイズ

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: osDisk: diskType:	GCP ディスクタイプ。	すべてのマシンのデフォルトのディスクタイプ。コントロールプレーンノードは、**pd-ssd** ディスクタイプを使用する必要があります。コンピュートノードは、**pd-ssd**、**pd-balance**、または **pd-standard** ディスクタイプを使用できます。
platform: gcp: defaultMachinePlatform: osImage: project:	オプション: デフォルトで、インストールプログラムは、コントロールプレーンおよびコンピュートマシンの起動に使用される RHCOS イメージをダウンロードしてインストールします。インストールプログラムが両方のタイプのマシンに使用するカスタム RHCOS イメージの場所を指定することで、デフォルトの動作をオーバーライドできます。	文字列。イメージが置かれている GCP プロジェクトの名前。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: osImage: name:	インストールプログラムがコントロールプレーンとコンピュートマシンの起動に使用するカスタム RHCOS イメージの名前。**platform.gcp.defaultMachinePlatform.osImage.project** を使用する場合、このフィールドは必須です。	文字列。RHCOS イメージの名前。
platform: gcp: defaultMachinePlatform: tags:	オプション: コントロールプレーンおよびコンピュートマシンに追加する別のネットワークタグ。	**network-tag1** などの 1 つ以上の文字列。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: type:	コントロールプレーンおよびコンピュートマシンの GCP マシンタイプ。	**n1-standard-4** などの GCP マシンタイプ。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: osDisk: encryptionKey: kmsKey: name:	マシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: osDisk: encryptionKey: kmsKey: keyRing:	KMS キーが属する Key Management Service (KMS) キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: osDisk: encryptionKey: kmsKey: location:	KMS キーリングが存在する GCP の場所。	GCP の場所。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: osDisk: encryptionKey: kmsKey: projectID:	KMS キーリングが存在するプロジェクトの ID。この値は、設定されていない場合、デフォルトで **platform.gcp.projectID** パラメーターの値になります。	GCP プロジェクト ID

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: osDisk: encryptionKey: kmsKeyServiceAccount:	コントロールプレーンおよびコンピュートマシンの暗号化要求に使用される GCP サービスアカウント。存在しない場合には、Compute Engine のデフォルトのサービスアカウントが使用されます。GCP サービスアカウントの詳細は、Google のドキュメントの service accounts を参照してください。	**<service_account_name>@<project_id>.iam.gserviceaccount.com** などの GCP サービスアカウントのメール。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: secureBoot:	クラスター内のすべてのマシンに Shielded VM セキュアブートを有効にするかどうか。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット保護などの追加のセキュリティープロトコルがあります。Shielded VM の詳細については、Shielded VM に関する Google のドキュメントを参照してください。	**Enabled** または **Disabled**。デフォルト値は、**Disabled** です。
platform: gcp: defaultMachinePlatform: confidentialCompute:	クラスター内のすべてのマシンに Confidential VM を使用するかどうか。Confidential VM は処理中のデータを暗号化します。Confidential Computing の詳細については、Confidential Computing に関する Google のドキュメントを参照してください。	**Enabled** または **Disabled**。デフォルト値は、**Disabled** です。

パラメーター	説明	値
platform: gcp: defaultMachinePlatform: onHostMaintenance:	ソフトウェアやハードウェアの更新など、ホストメンテナンスイベント中のすべての VM の動作を指定します。Confidential VM の場合は、このパラメーターを **Terminate** に設定する必要があります。Confidential VM はライブ VM 移行をサポートしていません。	**Terminate** または **Migrate**。デフォルト値は、**Migrate** です。

パラメーター	説明	値
controlPlane: platform: gcp: osDisk: encryptionKey: kmsKey: name:	コントロールプレーンマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。

パラメーター	説明	値
controlPlane: platform: gcp: osDisk: encryptionKey: kmsKey: keyRing:	コントロールプレーンマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
controlPlane: platform: gcp: osDisk: encryptionKey: kmsKey: location:	コントロールプレーンマシンの場合、キーリングが存在する GCP の場所。KMS の場所の詳細は、Google のドキュメント Cloud KMS locations を参照してください。	キーリングの GCP の場所。

パラメーター	説明	値
controlPlane: platform: gcp: osDisk: encryptionKey: kmsKey: projectID:	コントロールプレーンマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID

パラメーター	説明	値
controlPlane: platform: gcp: osDisk: encryptionKey: kmsKeyServiceAccount:	コントロールプレーンマシンの暗号化要求に使用される GCP サービスアカウント。存在しない場合には、Compute Engine のデフォルトのサービスアカウントが使用されます。GCP サービスアカウントの詳細は、Google のドキュメントの service accounts を参照してください。	**<service_account_name>@<project_id>.iam.gserviceaccount.com** などの GCP サービスアカウントのメール。

パラメーター	説明	値
controlPlane: platform: gcp: osDisk: diskSizeGB:	ディスクのサイズ (GB 単位)。この値はコントロールプレーンマシンに適用されます。	16 から 65536 までの整数。
controlPlane: platform: gcp: osDisk: diskType:	コントロールプレーンマシンの GCP ディスクタイプ。	コントロールプレーンマシンは、デフォルトの **pd-ssd** ディスクタイプを使用する必要があります。

パラメーター	説明	値
controlPlane: platform: gcp: tags:	オプション: コントロールプレーンマシンに追加する別のネットワークタグ。このパラメーターを設定すると、コントロールプレーンマシンの **platform.gcp.defaultMachinePlatform.tags** パラメーターが上書きされます。	**control-plane-tag1** などの 1 つ以上の文字列。
controlPlane: platform: gcp: type:	コントロールプレーン マシンの GCP マシンタイプ。設定されている場合、このパラメーターは **platform.gcp.defaultMachinePlatform.type** パラメーターを上書きします。	**n1-standard-4** などの GCP マシンタイプ。

パラメーター	説明	値
controlPlane: platform: gcp: zones:	インストールプログラムがコントロールプレーンマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。  **重要** GCP 64 ビット ARM インフラストラクチャーでクラスターを実行する場合は、Ampere Altra Arm CPU が利用可能なゾーンを使用するようにしてください。「GCP 可用性ゾーン」リンクで、64 ビット ARM プロセッサーと互換性のあるゾーンを確認できます。

パラメーター	説明	値
controlPlane: platform: gcp: secureBoot:	コントロールプレーンマシンの Shielded VM セキュアブートを有効にするかどうか。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット保護などの追加のセキュリティープロトコルがあります。Shielded VM の詳細については、Shielded VM に関する Google のドキュメントを参照してください。	**Enabled** または **Disabled**。デフォルト値は、**Disabled** です。
controlPlane: platform: gcp: confidentialCompute:	コントロールプレーンマシンの Confidential VM を有効にするかどうか。Confidential VM は処理中のデータを暗号化します。Confidential VM の詳細については、Confidential Computing に関する Google のドキュメントを参照してください。	**Enabled** または **Disabled**。デフォルト値は、**Disabled** です。

パラメーター	説明	値
controlPlane: platform: gcp: onHostMaintenance:	ソフトウェアやハードウェアの更新など、ホストメンテナンスイベント中のコントロールプレーン VM の動作を指定します。Confidential VM の場合は、このパラメーターを **Terminate** に設定する必要があります。Confidential VM はライブ VM 移行をサポートしていません。	**Terminate** または **Migrate**。デフォルト値は、**Migrate** です。
compute: platform: gcp: osDisk: encryptionKey: kmsKey: name:	コントロールマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。

パラメーター	説明	値
compute: platform: gcp: osDisk: encryptionKey: kmsKey: keyRing:	コンピュートマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
compute: platform: gcp: osDisk: encryptionKey: kmsKey: location:	コンピュートマシンの場合、キーリングが存在する GCP の場所。KMS の場所の詳細は、Google のドキュメント Cloud KMS locations を参照してください。	キーリングの GCP の場所。

パラメーター	説明	値
compute: platform: gcp: osDisk: encryptionKey: kmsKey: projectID:	コンピュートマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID

パラメーター	説明	値
compute: platform: gcp: osDisk: encryptionKey: kmsKeyServiceAccount:	コンピュートマシンの暗号化要求に使用される GCP サービスアカウント。この値が設定されていない場合には、Compute Engine のデフォルトのサービスアカウントが使用されます。GCP サービスアカウントの詳細は、Google のドキュメントの service accounts を参照してください。	**<service_account_name>@<project_id>.iam.gserviceaccount.com** などの GCP サービスアカウントのメール。

パラメーター	説明	値
compute: platform: gcp: osDisk: diskSizeGB:	ディスクのサイズ (GB 単位)。この値はコンピュートマシンに適用されます。	16 から 65536 までの整数。
compute: platform: gcp: osDisk: diskType:	コンピュートマシンの GCP ディスクタイプ。	**pd-ssd**、**pd-standard**、または **pd-balance**。デフォルトは **pd-ssd** です。

パラメーター	説明	値
compute: platform: gcp: tags:	オプション: コンピュートマシンに追加する別のネットワークタグ。このパラメーターを設定すると、コンピュートマシンの **platform.gcp.defaultMachinePlatform.tags** パラメーターが上書きされます。	**compute-network-tag1** などの 1 つ以上の文字列。
compute: platform: gcp: type:	コンピュートマシンの GCP マシンタイプ。設定されている場合、このパラメーターは **platform.gcp.defaultMachinePlatform.type** パラメーターを上書きします。	**n1-standard-4** などの GCP マシンタイプ。

パラメーター	説明	値
compute: platform: gcp: zones:	インストールプログラムがコンピュートマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。 **重要** GCP 64 ビット ARM インフラストラクチャーでクラスターを実行する場合は、Ampere Altra Arm CPU が利用可能なゾーンを使用するようにしてください。「GCP 可用性ゾーン」リンクで、64 ビット ARM プロセッサーと互換性のあるゾーンを確認できます。
compute: platform: gcp: secureBoot:	コンピュートマシン用に Shielded VM のセキュアブートを有効にするかどうか。Shielded VM には、セキュアブート、ファームウェアと整合性の監視、ルートキット保護などの追加のセキュリティープロトコルがあります。Shielded VM の詳細については、Shielded VM に関する Google のドキュメントを参照してください。	**Enabled** または **Disabled**。デフォルト値は、**Disabled** です。

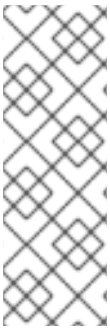
パラメーター	説明	値
compute: platform: gcp: confidentialCompute:	コンピューティングマシンの Confidential VM を有効にするかどうか。Confidential VM は処理中のデータを暗号化します。Confidential VM の詳細については、Confidential Computing に関する Google のドキュメントを参照してください。	**Enabled** または **Disabled**。デフォルト値は、**Disabled** です。
compute: platform: gcp: onHostMaintenance:	ソフトウェアやハードウェアの更新など、ホストメンテナンスイベント中のコンピューティング VM の動作を指定します。Confidential VM の場合は、このパラメーターを **Terminate** に設定する必要があります。Confidential VM はライブ VM 移行をサポートしていません。	**Terminate** または **Migrate**。デフォルト値は、**Migrate** です。

9.15. GCP でのクラスターのアンインストール

Google Cloud Platform (GCP) にデプロイしたクラスターを削除できます。

9.15.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。



注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。たとえば、一部の Google Cloud リソースには共有 VPC ホストプロジェクトで IAM パーミッション が必要になるか、削除する必要のあるヘルスチェック が使用されていない可能性があります。

前提条件

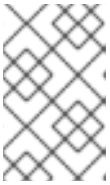
- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

9.15.2. Cloud Credential Operator ユーティリティーを使用した Google Cloud Platform リソースの削除

クラスターの外部で管理される短期認証情報を使用する OpenShift Container Platform クラスターをアンインストールした後、CCO ユーティリティー (**ccoctl**) を使用して、インストール中に **ccoctl** が作成した Google Cloud Platform (GCP) リソースを削除できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

- 短期認証情報を使用する GCP 上の OpenShift Container Platform クラスターをアンインストールします。

手順

1. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --to=<path_to_directory_for_credentials_requests> 2
```

1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

2 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

3. 次のコマンドを実行して、**ccoctl** が作成した GCP リソースを削除します。

```
$ ccoctl gcp delete \
  --name=<name> \ 1
  --project=<gcp_project_id> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --force-delete-custom-roles 3
```

1 **<name>** は、クラウドリソースを最初に作成してタグ付けするために使用された名前と一致します。

2 **<gcp_project_id>** は、クラウドリソースを削除する GCP プロジェクト ID です。

3 オプション: このパラメーターは、**ccoctl** ユーティリティーがインストール中に作成するカスタムロールを削除します。GCP は、カスタムロールを即座に完全削除するわけではありません。詳細は、カスタムロールの削除 に関する GCP ドキュメントを参照してください。

検証

- リソースが削除されたことを確認するには、GCP にクエリーを実行します。詳細については、GCP ドキュメントをご覧ください。

第10章 IBM CLOUD へのインストール

10.1. IBM CLOUD へのインストールの準備

このセクションに記載されているインストールワークフローは、IBM Cloud® インフラストラクチャー環境向けです。現時点では、IBM Cloud® Classic はサポートされていません。Classic インフラストラクチャーと VPC インフラストラクチャーの違いの詳細は、IBM® ドキュメント を参照してください。

10.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。

10.1.2. IBM Cloud に OpenShift Container Platform をインストールするための要件

OpenShift Container Platform を IBM Cloud ®にインストールする前に、サービスアカウントを作成し、IBM Cloud® アカウントを設定する必要があります。アカウントの作成、API サービスの有効化、DNS の設定、IBM Cloud® アカウント制限、およびサポートされる IBM Cloud® リージョンの詳細については、IBM Cloud® アカウントの設定 を参照してください。

クラスターを IBM Cloud® にインストールするときは、クラウドの認証情報を手動で管理する必要があります。これは、クラスターをインストールする前に、手動モードの Cloud Credential Operator (CCO) を設定して実行します。詳細については、IBM Cloud® 用の IAM の設定 を参照してください。

10.1.3. IBM Cloud に OpenShift Container Platform をインストールする方法の選択

インストーラーがプロビジョニングしたインフラストラクチャーを使用して、IBM Cloud® に OpenShift Container Platform をインストールできます。このプロセスでは、インストールプログラムを使用して、クラスターの基盤となるインフラストラクチャーをプロビジョニングします。現時点では、ユーザーによってプロビジョニングされたインフラストラクチャーを使用した IBM Cloud® への OpenShift Container Platform のインストールはサポートされていません。

インストーラーがプロビジョニングしたインストールプロセスの詳細については、インストールプロセス を参照してください。

10.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下のいずれかの方法を使用して、OpenShift Container Platform インストールプログラムによってプロビジョニングされた IBM Cloud ®インフラストラクチャーにクラスターをインストールできます。

- **カスタマイズされたクラスターの IBM Cloud® へのインストール**: インストールプログラムがプロビジョニングする IBM Cloud® インフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の多くのカスタマイズオプションは、インストール後 に利用できます。
- **ネットワークをカスタマイズして IBM Cloud® にクラスターをインストールする**: インストール中に OpenShift Container Platform ネットワーク設定をカスタマイズして、クラスターが既存の IP アドレス割り当てと共存し、ネットワーク要件に準拠できるようにすることができます。
- **IBM Cloud® 上の既存の VPC へのクラスターのインストール**: OpenShift Container Platform を既存の IBM Cloud® にインストールできます。このインストール方法は、新規アカウントまたは

インフラストラクチャーを作成する際の制限など、会社のガイドラインによる制約がある場合に使用できます。

- **既存の VPC へのプライベートクラスターの** インストール: 既存の Virtual Private Cloud (VPC) にプライベートクラスターをインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

- **ネットワークが制限された環境での IBM Cloud VPC へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、installer-provisioned infrastructure上の IBM Cloud VPC に OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。

10.1.4. 次のステップ

- IBM Cloud® アカウントの設定

10.2. IBM CLOUD アカウントの設定

OpenShift Container Platform をインストールする前に、IBM Cloud® アカウントを設定する必要があります。

10.2.1. 前提条件

- サブスクリプションのある IBM Cloud® アカウントを持っている。無料または試用版の IBM Cloud® アカウントに OpenShift Container Platform をインストールすることはできません。

10.2.2. IBM Cloud のクォータと制限

OpenShift Container Platform クラスターは多数の IBM Cloud® コンポーネントを使用し、デフォルトのクォータと制限は OpenShift Container Platform クラスターをインストールする機能に影響を与えます。特定のクラスター設定を使用する場合、特定のリージョンにクラスターをデプロイするか、アカウントから複数のクラスターを実行する場合は、IBM Cloud® アカウントに追加のリソースを要求する必要がある場合があります。

デフォルトの IBM Cloud® クォータとサービス制限の包括的なリストについては、IBM Cloud® のドキュメント Quotas and service limits を参照してください。

Virtual Private Cloud (VPC)
各 OpenShift Container Platform クラスターは、独自の VPC を作成します。リージョンごとの VPC のデフォルトのクォータは 10 で、10 個のクラスターを許可します。1 つのリージョンに 10 を超えるクラスターを含めるには、このクオータを増やす必要があります。

アプリケーションロードバランサー
デフォルトでは、各クラスターは 3 つのアプリケーションロードバランサー (ALB) を作成します。

- マスター API サーバーの内部ロードバランサー
- マスター API サーバーの外部ロードバランサー
- ルーターのロードバランサー

追加の **LoadBalancer** サービスオブジェクトを作成して、追加の ALB を作成できます。VPC ALB のデフォルトのクォータは、リージョンごとに 50 です。50 を超える ALB を使用するには、このクオータを増やす必要があります。

VPC ALB がサポートされています。従来の ALB は、IBM Cloud® ではサポートされていません。

フローティング IP アドレス
デフォルトでは、インストールプログラムは、コントロールプレーンとコンピューティングマシンをリージョン内のすべてのアベイラビリティーゾーンに分散して、高可用性設定でクラスターをプロビジョニングします。各アベイラビリティーゾーンで、パブリックゲートウェイが作成され、個別のフローティング IP アドレスが必要になります。

フローティング IP アドレスのデフォルトのクォータは、アベイラビリティーゾーンごとに 20 アドレスです。デフォルトのクラスター設定では、3 つのフローティング IP アドレスが生成されます。

- **us-east-1** プライマリーゾーンの 2 つのフローティング IP アドレス。ブートストラップノードに関連付けられている IP アドレスは、インストール後に削除されます。
- **us-east-2** セカンダリーゾーンの 1 つのフローティング IP アドレス。
- **us-east-3**セカンダリーゾーンの 1 つのフローティング IP アドレス。

IBM Cloud® は、アカウント内のリージョンごとに最大 19 個のクラスターをサポートできます。19 を超えるデフォルトクラスターを計画している場合は、このクオータを増やす必要があります。

Virtual Server Instances (VSI)
デフォルトでは、クラスターは **bx2-4x16** プロファイルを使用して VSI を作成します。これには、デフォルトで次のリソースが含まれます。

- 仮想 CPU 4 個
- 16 GB RAM

次のノードが作成されます。

- インストールの完了後に削除される 1 台の **bx2-4x16** ブートストラップマシン
- 3 つの **bx2-4x16** コントロールプレーンノード
- 3 つの **bx2-4x16** コンピュートノード

詳細については、IBM Cloud® のドキュメント supported profiles を参照してください。

表10.1 VSI コンポーネントのクォータと制限

VSI コンポーネント	デフォルトの IBM Cloud® クォータ	デフォルトのクラスター設定	クラスターの最大数
vCPU	リージョンごとに 200 の vCPU	28 の vCPU、またはブートストラップの削除後は 24 個の vCPU	リージョンごとに 8
RAM	リージョンあたり 1600 GB	112 GB、またはブートストラップの削除後は 96 GB	リージョンごとに 16
ストレージ	リージョンごとに 18 TB	1050 GB、またはブートストラップの削除後は 900 GB	リージョンごとに 19

VSI コンポーネント	デフォルトの IBM Cloud® クォータ	デフォルトのクラスター設定	クラスターの最大数

表に記載されているリソースを超える予定がある場合は、IBM Cloud® アカウントのクォータを増やす必要があります。

ブロックストレージボリューム
VPC マシンごとに、ブートボリューム用にブロックストレージデバイスが接続されます。デフォルトのクラスター設定では、7 台の VPC マシンが作成され、7 つのブロックストレージボリュームが作成されます。IBM Cloud® ストレージクラスの追加の Kubernetes 永続ボリューム要求 (PVC) は、追加のブロックストレージボリュームを作成します。VPC ブロックストレージボリュームのデフォルトのクォータは、リージョンごとに 300 です。300 を超えるボリュームを使用するには、このクオータを増やす必要があります。

10.2.3. DNS 解決の設定

DNS 解決の設定方法は、インストールする OpenShift Container Platform クラスターのタイプによって異なります。

- パブリッククラスターをインストールする場合は、IBM Cloud Internet Services (CIS) を使用します。
- プライベートクラスターをインストールする場合は、IBM Cloud® DNS サービス (DNS サービス) を使用します。

10.2.3.1. DNS 解決のための IBM Cloud Internet Services の使用

インストールプログラムは、IBM Cloud® Internet Services (CIS) を使用してクラスター DNS 解決を設定し、パブリッククラスターの名前検索を提供します。

注記

この製品は IPv6 をサポートしていないため、デュアルスタックまたは IPv6 環境は使用できません。

クラスターと同じアカウントの CIS にドメインゾーンを作成する必要があります。また、ゾーンがドメインに対して権限を持っていることを確認する必要があります。これは、root ドメインまたはサブドメインを使用して行うことができます。

前提条件

- IBM Cloud® CLI がインストールされている。
- 既存のドメインとレジストラがあります。詳細については、IBM® の ドキュメント を参照してください。

手順

1. クラスターで使用する CIS インスタンスを作成します。

a. CIS プラグインをインストールします。

```
$ ibmcloud plugin install cis
```

b. CIS インスタンスを作成します。

```
$ ibmcloud cis instance-create <instance_name> standard 1
```

1 CIS がクラスターサブドメインとその DNS レコードを管理するには、少なくとも **Standard** プランが必要です。

2. 既存のドメインを CIS インスタンスに接続します。

a. CIS のコンテキストインスタンスを設定します。

```
$ ibmcloud cis instance-set <instance_name> 1
```

1 インスタンスクラウドのリソース名。

b. CIS のドメインを追加します。

```
$ ibmcloud cis domain-add <domain_name> 1
```

1 完全修飾ドメイン名。設定する予定に応じて、ドメイン名として root ドメインまたはサブドメインのいずれかの値を使用できます。

注記

root ドメインは、**openshiftcorp.com** の形式を使用します。サブドメインは、**clusters.openshiftcorp.com** の形式を使用します。

3. CIS Web コンソール を開き、**Overview** ページに移動して、CIS ネームサーバーをメモします。これらのネームサーバーは、次のステップで使用されます。

4. ドメインのレジストラーまたは DNS プロバイダーでドメインまたはサブドメインのネームサーバーを設定します。詳細については、IBM Cloud® の ドキュメント を参照してください。

10.2.3.2. DNS 解決のための IBM Cloud DNS サービスの使用

インストールプログラムは、IBM Cloud® DNS サービスを使用してクラスター DNS 解決を設定し、プライベートクラスターの名前ルックアップを提供します。

クラスターの DNS サービスインスタンスを作成し、DNS サービスインスタンスに DNS ゾーンを追加して、DNS 解決を設定します。ゾーンがドメインに対して権限を持っていることを確認してください。これは、root ドメインまたはサブドメインを使用して行うことができます。

注記

IBM Cloud® は IPv6 をサポートしていないため、デュアルスタックまたは IPv6 環境は使用できません。

前提条件

- IBM Cloud® CLI がインストールされている。
- 既存のドメインとレジストラがあります。詳細については、IBM® の ドキュメント を参照してください。

手順

1. クラスターで使用する DNS サービスインスタンスを作成します。
   a. 次のコマンドを実行して、DNS サービスプラグインをインストールします。

```
$ ibmcloud plugin install cloud-dns-services
```

   b. 次のコマンドを実行して、DNS サービスインスタンスを作成します。

```
$ ibmcloud dns instance-create <instance-name> standard-dns 1
```

   1 DNS Services がクラスターサブドメインとその DNS レコードを管理するには、少なくとも **Standard** プランが必要です。

2. DNS サービスインスタンスの DNS ゾーンを作成します。
   a. 次のコマンドを実行して、ターゲットのオペレーティング DNS サービスインスタンスを設定します。

```
$ ibmcloud dns instance-target <instance-name>
```

   b. 次のコマンドを実行して、DNS サービスインスタンスに DNS ゾーンを追加します。

```
$ ibmcloud dns zone-create <zone-name> 1
```

   1 完全修飾ゾーン名。設定する予定に応じて、ゾーン名として root ドメインまたはサブドメインのいずれかの値を使用できます。root ドメインは、**openshiftcorp.com** の形式を使用します。サブドメインは、**clusters.openshiftcorp.com** の形式を使用します。

3. 作成した DNS ゾーンの名前を記録します。インストールプロセスの一環として、クラスターをデプロイする前に、**install-config.yaml** ファイルを更新する必要があります。DNS ゾーンの名前を **baseDomain** パラメーターの値として使用します。

注記

許可されたネットワークを管理したり、ADNS リソースレコードを設定したりする必要はありません。必要に応じて、インストールプログラムはこれらのリソースを自動的に設定します。

10.2.4. IBM Cloud IAM ポリシーと API キー

OpenShift Container Platform を IBMCloud® アカウントにインストールするには、インストールプログラムに IAM API キーが必要です。これにより、IBM Cloud® サービス API にアクセスするための認証と

認証が提供されます。必要なポリシーを含む既存の IAM API キーを使用するか、新しいポリシーを作成できます。

IBM Cloud® IAM の概要は、IBM Cloud® の ドキュメント を参照してください。

10.2.4.1. 必要なアクセスポリシー

必要なアクセスポリシーを IBM Cloud® アカウントに割り当てる必要があります。

表10.2 必要なアクセスポリシー

サービスのタイプ	サービス	アクセスポリシーの範囲	プラットフォームアクセス	サービスアクセス
アカウント管理	IAM ID サービス	すべてのリソースまたはリソースのサブセット[1]	エディター、Operator、ビューアー、管理者	サービス ID 作成者
アカウント管理 [2]	アイデンティティーおよびアクセス管理	すべてのリソース	エディター、Operator、ビューアー、管理者	
アカウント管理	リソースグループのみ	アカウント内のすべてのリソースグループ	Administrator	
IAM サービス	クラウドオブジェクトストレージ	すべてのリソースまたはリソースのサブセット[1]	エディター、Operator、ビューアー、管理者	リーダー、ライター、マネージャー、コンテンツリーダー、オブジェクトリーダー、オブジェクトライター
IAM サービス	インターネットサービス	すべてのリソースまたはリソースのサブセット[1]	エディター、Operator、ビューアー、管理者	リーダー、ライター、マネージャー
IAM サービス	DNS サービス	すべてのリソースまたはリソースのサブセット[1]	エディター、Operator、ビューアー、管理者	リーダー、ライター、マネージャー
IAM サービス	VPC インフラストラクチャーサービス	すべてのリソースまたはリソースのサブセット[1]	エディター、Operator、ビューアー、管理者	リーダー、ライター、マネージャー

1. ポリシーアクセススコープは、アクセスを割り当てる粒度に基づいて設定する必要があります。スコープは、**すべてのリソース** または **選択した属性に基づくリソース**に設定できます。
2. オプション: このアクセスポリシーは、インストールプログラムでリソースグループを作成する場合にのみ必要です。リソースグループの詳細については、IBM® の ドキュメント を参照してください。

10.2.4.2. アクセスポリシーの割り当て

IBM Cloud® IAM では、アクセスポリシーをさまざまなサブジェクトにアタッチできます。

- アクセスグループ (推奨)
- サービス ID
- User

推奨される方法は、アクセスグループ で IAM アクセスポリシーを定義することです。これにより、OpenShift Container Platform に必要なすべてのアクセスを整理し、ユーザーとサービス ID をこのグループにオンボードできます。必要に応じて、ユーザーとサービス ID に直接アクセスを割り当てることもできます。

10.2.4.3. API キーの作成

IBM Cloud® アカウントのユーザー API キーまたはサービス ID API キーを作成する必要があります。

前提条件

- 必要なアクセスポリシーを IBM Cloud® アカウントに割り当てている。
- IAM アクセスポリシーをアクセスグループまたはその他の適切なリソースにアタッチしている。

手順

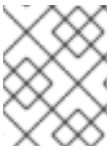
- IAM アクセスポリシーの定義方法に応じて、API キーを作成します。
たとえば、アクセスポリシーをユーザーに割り当てた場合は、ユーザー API キー を作成する必要があります。アクセスポリシーをサービス ID に割り当てた場合は、サービス ID API キー を作成する必要があります。アクセスポリシーがアクセスグループに割り当てられている場合は、どちらの API キータイプも使用できます。IBM Cloud® API キーの詳細は、Understanding API keys を参照してください。

10.2.5. サポートされている IBM Cloud リージョン

OpenShift Container Platform クラスターを以下のリージョンにデプロイできます。

- **au-syd** (Sydney, Australia)
- **br-sao** (Sao Paulo, Brazil)
- **ca-tor** (Toronto, Canada)
- **eu-de** (Frankfurt, Germany)
- **eu-gb** (London, United Kingdom)
- **eu-es** (Madrid, Spain)
- **jp-osa** (Osaka, Japan)
- **jp-tok** (Tokyo, Japan)
- **us-east** (Washington DC, United States)

- **us-south** (Dallas, United States)



注記

eu-es (Madrid, Spain) リージョンへのクラスターのデプロイは、OpenShift Container Platform 4.14.6 以前のバージョンではサポートされていません。

10.2.6. 次のステップ

- IBM Cloud® 用の IAM の設定

10.3. IBM CLOUD 用の IAM の設定

クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境では、クラスターのインストール前に Cloud Credential Operator (CCO) を手動モードに配置する必要があります。

10.3.1. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報を Kubernetes カスタムリソース定義 (CRD) として管理します。**credentialsMode** パラメーターの異なる値を **install-config.yaml** ファイルに設定し、組織のセキュリティー要件に応じて CCO を設定できます。

管理者レベルのクレデンシャルシークレットをクラスター **kube-system** プロジェクトに格納することは、IBM Cloud® ではサポートされていません。したがって、OpenShift Container Platform をインストールするときは、CCO の **credentialsMode** パラメーターを **Manual** に設定し、クラウドクレデンシャルを手動で管理する必要があります。

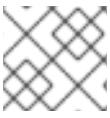
手動モードを使用すると、クラスターに管理者レベルの認証情報を保存する必要なく、各クラスターコンポーネントに必要なパーミッションのみを指定できます。お使いの環境でクラウドプロバイダーのパブリック IAM エンドポイントへの接続がない場合も、このモードを使用できます。ただし、各アップグレードについて、パーミッションを新規リリースイメージを使用して手動で調整する必要があります。また、それらを要求するすべてのコンポーネントについて認証情報を手動で指定する必要があります。

関連情報

- Cloud Credential Operator について

10.3.2. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。

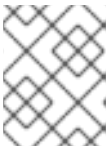
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
```

```
  aws          Manage credentials objects for AWS cloud
  azure        Manage credentials objects for Azure
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for IBM Cloud
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

関連情報

- IBM Cloud® の API キーのローテーション

10.3.3. 次のステップ

- カスタマイズを使用した IBM Cloud® へのクラスターのインストール

10.3.4. 関連情報

- 手動で維持された認証情報でクラスターを更新する準備

10.4. IBM CLOUD におけるユーザー管理の暗号化

デフォルトでは、OpenShift Container Platform クラスターのデプロイ時に以下を保護するために、プロバイダー管理の暗号化が使用されます。

- コントロールプレーンおよびコンピュートマシンのルート (ブート) ボリューム
- クラスターのデプロイ後にプロビジョニングされる永続ボリューム (データボリューム)

インストールプロセス中に IBM® Key Protect for IBM Cloud® (Key Protect) のルート鍵を指定することで、デフォルトの動作をオーバーライドできます。

独自のルート鍵を配置する場合、**encryptionKey** パラメーターを使用してルート鍵の Cloud Resource Name (CRN)を指定するために、インストール設定ファイル (**install-config.yaml**) を変更します。

以下を指定できます。

- すべてのクラスターマシンに同じルート鍵を使用します。そのためには、鍵をクラスターのデフォルトのマシン設定の一部として指定します。
  デフォルトのマシン設定の一部として指定されると、すべてのマネージドストレージクラスがこの鍵で更新されます。そのため、インストール後にプロビジョニングされるデータボリュームもこの鍵を使用して暗号化されます。
- コントロールプレーンとコンピュートマシンプールには別のルート鍵を使用します。

encryptionKey パラメーターの詳細は、追加の IBM Cloud 設定パラメーター を参照してください。

注記

Key Protect が IBM Cloud Block Storage サービスと統合されていることを確認してください。詳細は、Key Protect の ドキュメント を参照してください。

10.4.1. 次のステップ

OpenShift Container Platform クラスターをインストールします。

- カスタマイズを使用した IBM Cloud へのクラスターのインストール
- ネットワークをカスタマイズして IBM Cloud にクラスターをインストールする
- クラスターの IBM Cloud の既存 VPC へのインストール
- プライベートクラスターを IBM Cloud にインストールする

10.5. カスタマイズを使用した IBM CLOUD へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが IBM Cloud® 上にプロビジョニングするインフラストラクチャーに、カスタマイズしたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

10.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように IBM Cloud® アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細については、IBM Cloud® 用の IAM の設定 を参照してください。

10.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

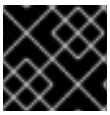
クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

10.5.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

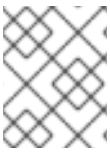
キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

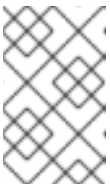
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

10.5.4. インストールプログラムの取得

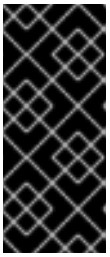
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

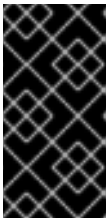
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

10.5.5. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

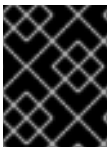
前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IC_API_KEY=<api_key>
```



重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

10.5.6. インストール設定ファイルの作成

IBM Cloud® にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットとするプラットフォームとしてibmcloudを選択します。

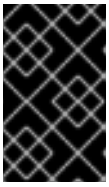
iii. クラスターをデプロイするリージョンを選択します。

iv. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

v. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

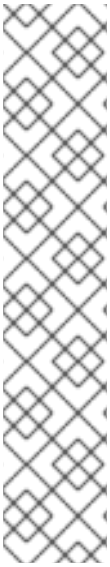
- IBM Cloud® のインストール設定パラメーター

10.5.6.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表10.3 最小リソース要件

マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

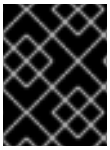
プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

10.5.6.2. IBM Cloud 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    ibmcloud: {}
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    ibmcloud: {}
  replicas: 3
metadata:
  name: test-cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
```

```
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (9)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ibmcloud:
    region: us-south (10)
credentialsMode: Manual
publish: External
pullSecret: '{"auths": ...}' (11)
fips: false (12)
sshKey: ssh-ed25519 AAAA... (13)
```

(1) (8) (10) (11) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(2) (5) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (6) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

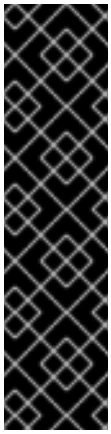
(4) (7) ハイパースレッディングとも呼ばれる同時マルチスレッドを有効または無効にします。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

(9) インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

(12) FIPS モードを有効または無効にします。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。



重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

13 オプション: クラスター内のマシンにアクセスするのに使用する **sshKey** 値をオプションで指定できます。

注記

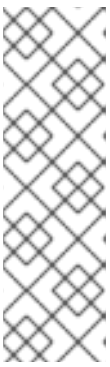
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

10.5.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

10.5.7. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

   ```
   apiVersion: v1
   baseDomain: cluster1.example.com
   credentialsMode: Manual 1
   compute:
   - architecture: amd64
     hyperthreading: Enabled
   ```

   1 この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
```

```
--credentials-requests-dir=<path_to_credential_requests_directory> \ 1
--name=<cluster_name> \ 2
--output-dir=<installation_directory> \ 3
--resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

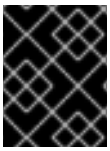
```
$ grep resourceGroupName <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

10.5.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

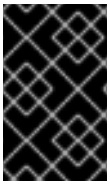
```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

10.5.9. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

10.5.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

10.5.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 し

て、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

10.5.12. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

10.6. ネットワークをカスタマイズして IBM CLOUD にクラスターをインストールする

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが IBM Cloud® 上にプロビジョニングするインフラストラクチャーに、カスタマイズしたネットワーク設定でクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

10.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように IBM Cloud® アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細については、IBM Cloud® 用の IAM の設定 を参照してください。

10.6.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

10.6.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

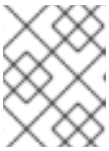
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

10.6.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

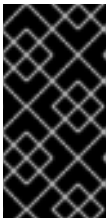
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

10.6.5. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IC_API_KEY=<api_key>
```

重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

10.6.6. インストール設定ファイルの作成

IBM Cloud® にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットとするプラットフォームとして**ibmcloud**を選択します。

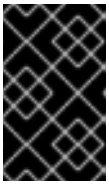
iii. クラスターをデプロイするリージョンを選択します。

iv. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

v. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

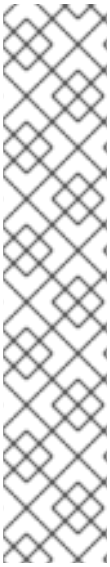
- IBM Cloud® のインストール設定パラメーター

10.6.6.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表10.4 最小リソース要件

マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

10.6.6.2. IBM Cloud 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    ibmcloud: {}
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    ibmcloud: {}
  replicas: 3
metadata:
  name: test-cluster (8)
networking: (9)
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
```

```
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (10)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ibmcloud:
    region: us-south (11)
credentialsMode: Manual
publish: External
pullSecret: '{"auths": ...}' (12)
fips: false (13)
sshKey: ssh-ed25519 AAAA... (14)
```

(1) (8) (11) (12) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(2) (5) (9) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (6) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(4) (7) ハイパースレッディングとも呼ばれる同時マルチスレッドを有効または無効にします。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

(10) インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

(13) FIPS モードを有効または無効にします。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

14 オプション: クラスター内のマシンにアクセスするのに使用する **sshKey** 値をオプションで指定できます。

注記

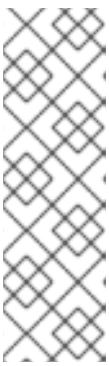
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

10.6.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

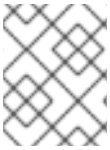
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

10.6.7. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

   ```
   apiVersion: v1
   baseDomain: cluster1.example.com
   credentialsMode: Manual 1
   compute:
   - architecture: amd64
     hyperthreading: Enabled
   ```

   1 この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
```

```
--credentials-requests-dir=<path_to_credential_requests_directory> \ 1
--name=<cluster_name> \ 2
--output-dir=<installation_directory> \ 3
--resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

```
$ grep resourceGroupName <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

10.6.8. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。

注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

重要

CIDR 範囲 **172.17.0.0/16** は libVirt によって予約されています。この範囲、またはこの範囲と重複する範囲をクラスター内のネットワークに使用することはできません。

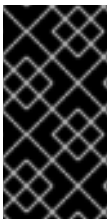
フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 でネットワークプラグインをさらにカスタマイズできます。

10.6.9. 高度なネットワーク設定の指定

ネットワークプラグインに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。



重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
```

```
  name: cluster
spec:
```

3. 次の例のように、**cluster-network-03-config.yml** ファイルでクラスターの高度なネットワーク設定を指定します。

   OVN-Kubernetes ネットワークプロバイダーを有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

10.6.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

10.6.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表10.5 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

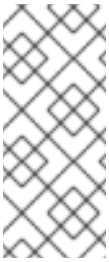
フィールド	型	説明
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定

defaultNetwork オブジェクトの値は、以下の表で定義されます。

表10.6 **defaultNetwork** オブジェクト

フィールド	型	説明

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表10.7 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。

フィールド	型	説明
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表10.8 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。

フィールド	型	説明
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表10.9 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表10.10 **policyAuditConfig オブジェクト**

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。

フィールド	型	説明
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表10.11 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

フィールド	型	説明
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表10.12 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表10.13 **gatewayConfig.ipv6** object

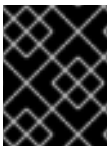
フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表10.14 **ipsecConfig** オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```



重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表10.15 **kubeProxyConfig** オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: `kubeProxyConfig:` `  proxyArguments:` `  iptables-min-sync-period:` `  - 0s`

10.6.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

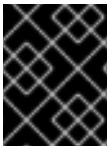
```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

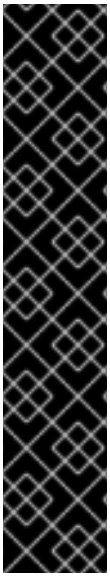


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

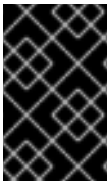


重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

10.6.12. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

10.6.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

10.6.14. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 し

て、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

10.6.15. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

10.7. クラスターの IBM CLOUD の既存 VPC へのインストール

OpenShift Container Platform バージョン 4.16 では、IBM Cloud® 上の既存の Virtual Private Cloud (VPC) にクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

10.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように IBM Cloud® アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細については、IBM Cloud® 用の IAM の設定 を参照してください。

10.7.2. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、既存の IBM® Virtual Private Cloud (VPC) のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを認識できないため、サブネットの CIDR などを選択できません。クラスターをインストールするサブネットのネットワークを設定する必要があります。

10.7.2.1. VPC を使用するための要件

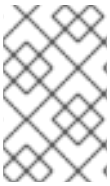
クラスターをインストールする前に、既存の VPC およびそのサブネットを適切に設定する必要があります。インストールプログラムでは、次のコンポーネントは作成されません。

- NAT ゲートウェイ
- サブネット

- ルートテーブル
- VPC ネットワーク

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化します。
- サブネットのルートテーブルを設定します。
- DHCP などの VPC オプションの設定



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

10.7.2.2. VPC 検証

VPC とすべてのサブネットは、既存のリソースグループ内にある必要があります。クラスターは既存の VPC にデプロイされます。

インストールの一環として、**install-config.yaml** ファイルで以下を指定します。

- VPC とサブネットを含む既存のリソースグループの名前 (**networkResourceGroupName**)
- 既存の VPC の名前 (**vpcName**)
- コントロールプレーンマシンおよびコンピュートマシン用に作成したサブネット (**controlPlaneSubnets** および **computeSubnets**)

注記

インストーラーによってプロビジョニングされる追加のクラスターリソースは、別のリソースグループ (**resourceGroupName**) にデプロイされます。クラスターをインストールする前に、このリソースグループを指定できます。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下を確認します。

- 指定したサブネットがすべて存在します。
- リージョン内の各アベイラビリティーゾーンに、以下を指定します。
  - コントロールプレーンマシンの 1 つのサブネット。
  - コンピュートマシン用に 1 つのサブネット。
- 指定したマシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれます。

注記

サブネット ID はサポートされていません。

10.7.2.3. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体で許可されます。
- TCP ポート 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

10.7.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

10.7.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

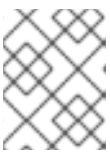
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

■

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

10.7.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

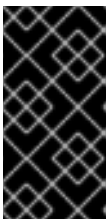
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

10.7.6. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

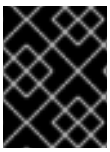
前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IC_API_KEY=<api_key>
```



重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

10.7.7. インストール設定ファイルの作成

IBM Cloud® にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットとするプラットフォームとしてibmcloudを選択します。

      iii. クラスターをデプロイするリージョンを選択します。

      iv. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

      v. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- IBM Cloud® のインストール設定パラメーター

10.7.7.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表10.16 最小リソース要件

マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

10.7.7.2. IBM Cloud 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    ibmcloud: {}
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    ibmcloud: {}
  replicas: 3
metadata:
  name: test-cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (10)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ibmcloud:
    region: eu-gb (11)
    resourceGroupName: eu-gb-example-cluster-rg (12)
    networkResourceGroupName: eu-gb-example-existing-network-rg (13)
    vpcName: eu-gb-example-network-1 (14)
    controlPlaneSubnets: (15)
      - eu-gb-example-network-1-cp-eu-gb-1
      - eu-gb-example-network-1-cp-eu-gb-2
      - eu-gb-example-network-1-cp-eu-gb-3
    computeSubnets: (16)
      - eu-gb-example-network-1-compute-eu-gb-1
      - eu-gb-example-network-1-compute-eu-gb-2
      - eu-gb-example-network-1-compute-eu-gb-3
credentialsMode: Manual
publish: External
pullSecret: '{"auths": ...}' (17)
fips: false (18)
sshKey: ssh-ed25519 AAAA... (19)
```

1 8 11 17 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 6 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 7 ハイパースレッディングとも呼ばれる同時マルチスレッドを有効または無効にします。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

9 マシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれている必要があります。

10 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 既存のリソースグループの名前。インストーラーによってプロビジョニングされたすべてのクラスターリソースは、このリソースグループにデプロイされます。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

13 既存の Virtual Private Cloud (VPC) を含むリソースグループの名前を指定します。既存の VPC およびサブネットはこのリソースグループにある必要があります。クラスターはこの VPC にインストールされます。

14 既存 VPC の名前を指定します。

15 コントロールプレーンマシンをデプロイする既存のサブネット名を指定します。サブネットは、指定した VPC に属している必要があります。リージョン内の各アベイラビリティーゾーンのサブネットを指定します。

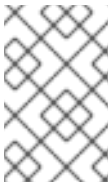
16 コンピュートマシンをデプロイする既存のサブネット名を指定します。サブネットは、指定した VPC に属している必要があります。リージョン内の各アベイラビリティーゾーンのサブネットを指定します。

18 FIPS モードを有効または無効にします。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

19 オプション: クラスター内のマシンにアクセスするのに使用する **sshKey** 値をオプションで指定できます。



注記

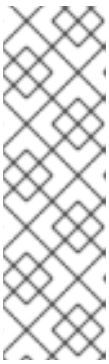
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

10.7.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

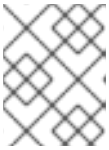
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
```

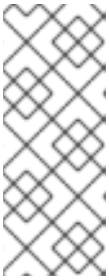
```
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

10.7.8. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

   ```
   apiVersion: v1
   baseDomain: cluster1.example.com
   credentialsMode: Manual (1)
   compute:
   - architecture: amd64
     hyperthreading: Enabled
   ```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
   ```

```
--included \ 1
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
--to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
  --credentials-requests-dir=<path_to_credential_requests_directory> \ 1
  --name=<cluster_name> \ 2
```

```
--output-dir=<installation_directory> \ 3
--resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

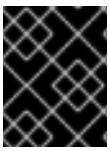
```
$ grep resourceGroupName <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

10.7.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

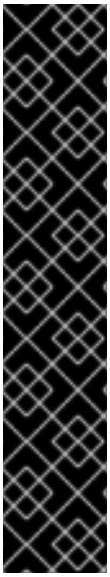
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

10.7.10. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

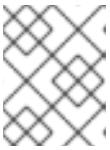
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

10.7.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

10.7.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 し

て、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

10.7.13. 次のステップ

- クラスターをカスタマイズ します。
- オプション: リモートヘルスレポートのオプトアウト

10.8. プライベートクラスターを IBM CLOUD にインストールする

OpenShift Container Platform バージョン 4.16 では、プライベートクラスターを既存の VPC にインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

10.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように IBM Cloud® アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細については、IBM Cloud® 用の IAM の設定 を参照してください。

10.8.2. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。
- IBM Cloud® DNS Services を使用して DNS ゾーンを作成し、それをクラスターの基本ドメインとして指定します。詳しくは、IBM Cloud® DNS サービスを使用して DNS 解決を設定するを参照してください。
- 以下にアクセスできるマシンからデプロイ。
  - プロビジョニングするクラウドの API サービス。
  - プロビジョニングするネットワーク上のホスト。
  - インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

10.8.3. IBM Cloud® 内のプライベートクラスター

IBM Cloud® 上にプライベートクラスターを作成するには、既存のプライベート VPC とサブネットを提供してクラスターをホストする必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

クラスターは、IBM Cloud® API にアクセスするために引き続きインターネットにアクセスする必要があります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックネットワークロードバランサー
- クラスターの **baseDomain** に一致するパブリック DNS ゾーン

インストールプログラムは、プライベート DNS ゾーンおよびクラスターに必要なレコードを作成するために指定する **baseDomain** を使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

10.8.3.1. 制限事項

IBM Cloud® 上のプライベートクラスターには、クラスターのデプロイメントに使用された既存の VPC に関連する制限のみが適用されます。

10.8.4. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、既存の IBM® Virtual Private Cloud (VPC) のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを認識できないため、サブネットの CIDR などを選択できません。クラスターをインストールするサブネットのネットワークを設定する必要があります。

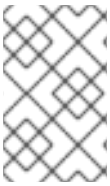
10.8.4.1. VPC を使用するための要件

クラスターをインストールする前に、既存の VPC およびそのサブネットを適切に設定する必要があります。インストールプログラムでは、次のコンポーネントは作成されません。

- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC ネットワーク

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化します。
- サブネットのルートテーブルを設定します。
- DHCP などの VPC オプションの設定



注記

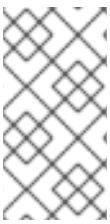
インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

10.8.4.2. VPC 検証

VPC とすべてのサブネットは、既存のリソースグループ内にある必要があります。クラスターは既存の VPC にデプロイされます。

インストールの一環として、**install-config.yaml** ファイルで以下を指定します。

- VPC とサブネットを含む既存のリソースグループの名前 (**networkResourceGroupName**)
- 既存の VPC の名前 (**vpcName**)
- コントロールプレーンマシンおよびコンピュートマシン用に作成したサブネット (**controlPlaneSubnets** および **computeSubnets**)



注記

インストーラーによってプロビジョニングされる追加のクラスターリソースは、別のリソースグループ (**resourceGroupName**) にデプロイされます。クラスターをインストールする前に、このリソースグループを指定できます。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下を確認します。

- 指定したサブネットがすべて存在します。

- リージョン内の各アベイラビリティーゾーンに、以下を指定します。
  - コントロールプレーンマシンの 1 つのサブネット。
  - コンピュートマシン用に 1 つのサブネット。
- 指定したマシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれます。

注記

サブネット ID はサポートされていません。

10.8.4.3. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体で許可されます。
- TCP ポート 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

10.8.5. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

10.8.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

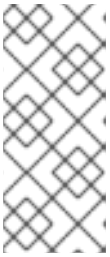
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

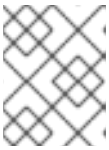
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、**~/.ssh/id_rsa** および **~/.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

10.8.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、クラウドネットワーク上の踏み台ホスト、または VPN 経由でネットワークにアクセスできるマシンにインストールファイルをダウンロードします。

プライベートクラスターのインストール要件の詳細については、プライベートクラスターを参照してください。

前提条件

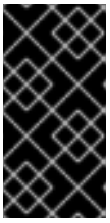
- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

10.8.8. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

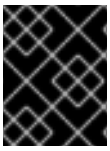
前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IC_API_KEY=<api_key>
```



重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

10.8.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

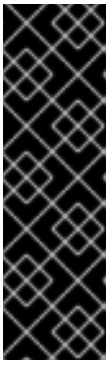
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

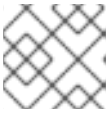
```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Cloud® のインストール設定パラメーター

10.8.9.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表10.17 最小リソース要件

マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

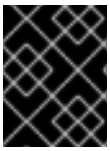
プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

10.8.9.2. IBM Cloud 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    ibmcloud: {}
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    ibmcloud: {}
  replicas: 3
metadata:
  name: test-cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16 (10)
  networkType: OVNKubernetes (11)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ibmcloud:
    region: eu-gb (12)
    resourceGroupName: eu-gb-example-cluster-rg (13)
    networkResourceGroupName: eu-gb-example-existing-network-rg (14)
    vpcName: eu-gb-example-network-1 (15)
    controlPlaneSubnets: (16)
      - eu-gb-example-network-1-cp-eu-gb-1
      - eu-gb-example-network-1-cp-eu-gb-2
      - eu-gb-example-network-1-cp-eu-gb-3
    computeSubnets: (17)
      - eu-gb-example-network-1-compute-eu-gb-1
      - eu-gb-example-network-1-compute-eu-gb-2
      - eu-gb-example-network-1-compute-eu-gb-3
credentialsMode: Manual
publish: Internal (18)
pullSecret: '{"auths": ...}' (19)
fips: false (20)
sshKey: ssh-ed25519 AAAA... (21)
```

(1) (8) (12) (19) 必須。

(2) (5) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (6) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1

つのコントロールプレーンプールのみが使用されます。

4 7 ハイパースレッディングとも呼ばれる同時マルチスレッドを有効または無効にします。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

9 マシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれている必要があります。

10 CIDR には、**platform.ibmcloud.controlPlaneSubnets** および **platform.ibmcloud.computeSubnets** で定義されたサブネットが含まれている必要があります。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

13 既存のリソースグループの名前。インストーラーによってプロビジョニングされたすべてのクラスターリソースは、このリソースグループにデプロイされます。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

14 既存の Virtual Private Cloud (VPC) を含むリソースグループの名前を指定します。既存の VPC およびサブネットはこのリソースグループにある必要があります。クラスターはこの VPC にインストールされます。

15 既存 VPC の名前を指定します。

16 コントロールプレーンマシンをデプロイする既存のサブネット名を指定します。サブネットは、指定した VPC に属している必要があります。リージョン内の各アベイラビリティーゾーンのサブネットを指定します。

17 コンピュートマシンをデプロイする既存のサブネット名を指定します。サブネットは、指定した VPC に属している必要があります。リージョン内の各アベイラビリティーゾーンのサブネットを指定します。

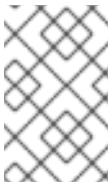
18 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。**publish** を **internal** に設定して、限定公開クラスターをデプロイします。デフォルト値は **External** です。

20 FIPS モードを有効または無効にします。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

21 オプション: クラスター内のマシンにアクセスするのに使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

10.8.9.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
```

```
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

10.8.10. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

   ```
   apiVersion: v1
   baseDomain: cluster1.example.com
   credentialsMode: Manual (1)
   compute:
   - architecture: amd64
     hyperthreading: Enabled
   ```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
   ```

```
--included \ 1
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
--to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
  --credentials-requests-dir=<path_to_credential_requests_directory> \ 1
  --name=<cluster_name> \ 2
```

```
--output-dir=<installation_directory> \ 3
--resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

```
$ grep resourceGroupName <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

10.8.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

10.8.12. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

10.8.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

10.8.14. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 し

て、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

10.8.15. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

10.9. ネットワークが制限された環境での IBM CLOUD へのクラスターのインストール

OpenShift Container Platform 4.16 では、IBM Cloud® 上の既存の Virtual Private Cloud (VPC) にアクセスできるインストールリリースコンテンツの内部ミラーを作成することで、制限されたネットワーク内にクラスターをインストールできます。

10.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターをホストするように IBM Cloud アカウントを設定 している。
- ネットワークが制限された環境およびインターネットにアクセスできるコンテナーイメージレジストリーがある。コンテナーイメージレジストリーは、OpenShift イメージレジストリーの内容をミラーリングし、インストールメディアを含むものである必要があります。詳細は、oc-mirror プラグインを使用した非接続インストールのイメージのミラーリング を参照してください。
- IBM Cloud® 上に以下の要件を満たす既存の VPC がある。
    - VPC に、ミラーレジストリーが含まれているか、他の場所でホストされているミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続がある。
    - VPC が、パブリックエンドポイントを使用して IBM Cloud® サービスエンドポイントにアクセスできる。ネットワーク制限によってパブリックサービスエンドポイントへのアクセスが制限されている場合は、エンドポイントのサービスを評価して、利用可能な代替エンドポイントを探してください。詳細は、IBM サービスエンドポイントへのアクセス を参照してください。

    インストールプログラムによってデフォルトでプロビジョニングされる VPC は使用できません。
- IBM Cloud® Virtual Private Endpoints を使用するようにエンドポイントゲートウェイを設定することを計画している場合は、以下の要件を考慮してください。
    - エンドポイントゲートウェイのサポートは、現在 **us-east** および **us-south** リージョンに限定されています。
    - VPC が、エンドポイントゲートウェイとの間のトラフィックを許可する必要があります。VPC のデフォルトのセキュリティーグループまたは新しいセキュリティーグループを使用して、ポート 443 でトラフィックを許可できます。詳細は、エンドポイントゲートウェイ

のトラフィックの許可 を参照してください。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細は、IBM Cloud VPC 用の IAM の設定 を参照してください。

10.9.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

10.9.2.1. 必要なインターネットアクセスとインストールホスト

インストールは、インターネットと制限されたネットワークの両方にアクセスできる踏み台ホストまたはポータブルデバイスを使用して実行します。インターネットにアクセスできるホストを使用して、以下を行う必要があります。

- インストールプログラム、OpenShift CLI (**oc**)、および CCO ユーティリティー (**ccoctl**) をダウンロードします。
- インストールプログラムを使用して Red Hat Enterprise Linux CoreOS (RHCOS) イメージを見つけ、インストール設定ファイルを作成します。
- **oc** を使用して、CCO コンテナーイメージから **ccoctl** を抽出します。
- **oc** および **ccoctl** を使用して、IBM Cloud® 用に IAM を設定します。

10.9.2.2. ミラーレジストリーへのアクセス

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。

このレジストリーは、ネットワークが制限された環境とインターネットの両方にアクセスできるミラーホスト上に作成することも、組織のセキュリティー制限に対応する他の方法を使用して作成することもできます。

非接続インストールのイメージをミラーリングする方法の詳細は、「関連情報」を参照してください。

10.9.2.3. IBM サービスエンドポイントへのアクセス

インストールプログラムは、以下の IBM Cloud® サービスエンドポイントへのアクセスを必要とします。

- Cloud Object Storage
- DNS Services
- Global Search
- Global Tagging

- Identity Services
- Resource Controller
- Resource Manager
- VPC

注記

インストールプロセス中に IBM® Key Protect for IBM Cloud® のルート鍵を指定する場合は、Key Protect のサービスエンドポイントも必要です。

デフォルトでは、サービスへのアクセスにパブリックエンドポイントが使用されます。ネットワーク制限によってパブリックサービスエンドポイントへのアクセスが制限されている場合は、デフォルトの動作をオーバーライドできます。

クラスターをデプロイする前に、インストール設定ファイル (**install-config.yaml**) を更新して、代替サービスエンドポイントの URI を指定できます。使用方法の詳細は、「関連情報」を参照してください。

10.9.2.4. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

関連情報

- oc-mirror プラグインを使用した非接続インストールのイメージのミラーリング
- 追加の IBM Cloud 設定パラメーター

10.9.3. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、既存の IBM® Virtual Private Cloud (VPC) のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを認識できないため、サブネットの CIDR などを選択できません。クラスターをインストールするサブネットのネットワークを設定する必要があります。

10.9.3.1. VPC を使用するための要件

クラスターをインストールする前に、既存の VPC およびそのサブネットを適切に設定する必要があります。インストールプログラムでは、次のコンポーネントは作成されません。

- NAT ゲートウェイ

- サブネット
- ルートテーブル
- VPC ネットワーク

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化します。
- サブネットのルートテーブルを設定します。
- DHCP などの VPC オプションの設定

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

10.9.3.2. VPC 検証

VPC とすべてのサブネットは、既存のリソースグループ内にある必要があります。クラスターは既存の VPC にデプロイされます。

インストールの一環として、**install-config.yaml** ファイルで以下を指定します。

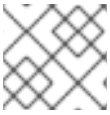
- VPC とサブネットを含む既存のリソースグループの名前 (**networkResourceGroupName**)
- 既存の VPC の名前 (**vpcName**)
- コントロールプレーンマシンおよびコンピュートマシン用に作成したサブネット (**controlPlaneSubnets** および **computeSubnets**)

注記

インストーラーによってプロビジョニングされる追加のクラスターリソースは、別のリソースグループ (**resourceGroupName**) にデプロイされます。クラスターをインストールする前に、このリソースグループを指定できます。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下を確認します。

- 指定したサブネットがすべて存在します。
- リージョン内の各アベイラビリティーゾーンに、以下を指定します。
  - コントロールプレーンマシンの 1 つのサブネット。
  - コンピュートマシン用に 1 つのサブネット。
- 指定したマシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれます。



注記

サブネット ID はサポートされていません。

10.9.3.3. クラスター間の分離

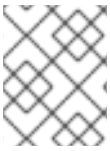
OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体で許可されます。
- TCP ポート 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

10.9.3.4. エンドポイントゲートウェイのトラフィックの許可

IBM Cloud® Virtual Private Endpoints を使用する場合は、エンドポイントゲートウェイとの間のトラフィックを許可するように Virtual Private Cloud (VPC) を設定する必要があります。

VPC のデフォルトのセキュリティーグループは、エンドポイントゲートウェイへのすべての送信トラフィックを許可するように設定されています。したがって、VPC とエンドポイントゲートウェイ間のトラフィックを許可する最も簡単な方法は、ポート 443 で受信トラフィックを許可するようにデフォルトのセキュリティーグループを変更することです。



注記

新しいセキュリティーグループを設定する場合は、受信トラフィックと送信トラフィックの両方を許可するようにセキュリティーグループを設定する必要があります。

前提条件

- IBM Cloud® コマンドラインインターフェイスユーティリティー (**ibmcloud**) がインストールされている。

手順

1. 次のコマンドを実行して、デフォルトのセキュリティーグループの識別子を取得します。

```
$ DEFAULT_SG=$(ibmcloud is vpc <your_vpc_name> --output JSON | jq -r
'.default_security_group.id')
```

2. 次のコマンドを実行して、ポート 443 で受信トラフィックを許可するルールを追加します。

```
$ ibmcloud is security-group-rule-add $DEFAULT_SG inbound tcp --remote 0.0.0.0/0 --port-
min 443 --port-max 443
```

注記

エンドポイントゲートウェイがこのセキュリティーグループを使用するように設定されていることを確認してください。

10.9.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

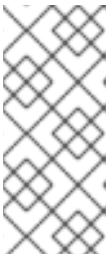
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

10.9.5. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

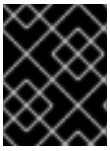
前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IC_API_KEY=<api_key>
```



重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

10.9.6. RHCOS クラスターイメージのダウンロード

インストールプログラムは、クラスターをインストールするために Red Hat Enterprise Linux CoreOS (RHCOS) イメージを必要とします。必要に応じて、デプロイ前に Red Hat Enterprise Linux CoreOS (RHCOS) をダウンロードすると、クラスターの作成時にインターネットにアクセスする必要がなくなります。

インストールプログラムを使用して、Red Hat Enterprise Linux CoreOS (RHCOS) イメージを見つけてダウンロードします。

前提条件

- インストールプログラムを実行しているホストがインターネットにアクセスできる。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install coreos print-stream-json
```

2. コマンドの出力を使用して、IBM Cloud® イメージの場所を見つけます。

```
.Example output
----
  "release": "415.92.202311241643-0",
  "formats": {
    "qcow2.gz": {
      "disk": {
        "location": "https://rhcos.mirror.openshift.com/art/storage/prod/streams/4.15-
9.2/builds/415.92.202311241643-0/x86_64/rhcos-415.92.202311241643-0-
ibmcloud.x86_64.qcow2.gz",
        "sha256":
"6b562dee8431bec3b93adeac1cfefcd5e812d41e3b7d78d3e28319870ffc9eae",
        "uncompressed-sha256":
"5a0f9479505e525a30367b6a6a6547c86a8f03136f453c1da035f3aa5daa8bc9"
----
```

3. イメージアーカイブをダウンロードして展開します。インストールプログラムがクラスターの作成に使用するホスト上でイメージを使用できるようにします。

10.9.7. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。
- レジストリーをミラーリングしたときに作成された **imageContentSourcePolicy.yaml** ファイルを用意する。
- ミラーレジストリーの証明書の内容を取得している。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

サンプルテンプレートをカスタマイズするときは、ネットワークが制限された環境でのインストールに必要な情報を必ず指定してください。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. 親の **platform.ibmcloud** フィールドで、クラスターをインストールする VPC のネットワークとサブネットを定義します。

```
vpcName: <existing_vpc>
controlPlaneSubnets: <control_plane_subnet>
computeSubnets: <compute_subnet>
```

platform.ibmcloud.vpcName には、既存の IBM Cloud VPC の名前を指定します。**platform.ibmcloud.controlPlaneSubnets** および **platform.ibmcloud.computeSubnets** には、それぞれコントロールプレーンマシンおよびコンピュートマシンをデプロイする既存のサブネットを指定します。

d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、レジストリーをミラーリングしたときに作成された **imageContentSourcePolicy.yaml** ファイルを使用します。

e. ネットワーク制限により、必要な IBM Cloud® サービスにアクセスするためのパブリックエンドポイントの使用が制限されている場合は、**serviceEndpoints** スタンザを **platform.ibmcloud** に追加して、代替サービスエンドポイントを指定します。

注記

指定できる代替サービスエンドポイントは、各サービスに 1 つだけです。

代替サービスエンドポイントの使用例

```
# ...
serviceEndpoints:
  - name: IAM
    url: <iam_alternate_endpoint_url>
  - name: VPC
    url: <vpc_alternate_endpoint_url>
  - name: ResourceController
```

```
    url: <resource_controller_alternate_endpoint_url>
  - name: ResourceManager
    url: <resource_manager_alternate_endpoint_url>
  - name: DNSServices
    url: <dns_services_alternate_endpoint_url>
  - name: COS
    url: <cos_alternate_endpoint_url>
  - name: GlobalSearch
    url: <global_search_alternate_endpoint_url>
  - name: GlobalTagging
    url: <global_tagging_alternate_endpoint_url>
# ...
```

f. オプション: パブリッシュストラテジーを **Internal** に設定します。

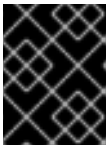
```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

注記

デフォルト値の **External** を使用する場合、IBM Cloud® Internet Services (CIS) のパブリックエンドポイントにネットワークがアクセスできる必要があります。CIS は Virtual Private Endpoints に対して有効ではありません。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Cloud® のインストール設定パラメーター

10.9.7.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表10.18 最小リソース要件

マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300

マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
Compute	RHCOS	2	8 GB	100 GB	300

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

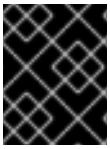
- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

10.9.7.2. IBM Cloud 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    ibm-cloud: {}
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    ibmcloud: {}
  replicas: 3
metadata:
  name: test-cluster (8)
networking:
```

```
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16 10
  networkType: OVNKubernetes 11
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ibmcloud:
    region: us-east 12
    resourceGroupName: us-east-example-cluster-rg 13
    serviceEndpoints: 14
      - name: IAM
        url: https://private.us-east.iam.cloud.ibm.com
      - name: VPC
        url: https://us-east.private.iaas.cloud.ibm.com/v1
      - name: ResourceController
        url: https://private.us-east.resource-controller.cloud.ibm.com
      - name: ResourceManager
        url: https://private.us-east.resource-controller.cloud.ibm.com
      - name: DNSServices
        url: https://api.private.dns-svcs.cloud.ibm.com/v1
      - name: COS
        url: https://s3.direct.us-east.cloud-object-storage.appdomain.cloud
      - name: GlobalSearch
        url: https://api.private.global-search-tagging.cloud.ibm.com
      - name: GlobalTagging
        url: https://tags.private.global-search-tagging.cloud.ibm.com
    networkResourceGroupName: us-east-example-existing-network-rg 15
    vpcName: us-east-example-network-1 16
    controlPlaneSubnets: 17
      - us-east-example-network-1-cp-us-east-1
      - us-east-example-network-1-cp-us-east-2
      - us-east-example-network-1-cp-us-east-3
    computeSubnets: 18
      - us-east-example-network-1-compute-us-east-1
      - us-east-example-network-1-compute-us-east-2
      - us-east-example-network-1-compute-us-east-3
credentialsMode: Manual
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 19
fips: false 20
sshKey: ssh-ed25519 AAAA... 21
additionalTrustBundle: | 22
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: 23
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 8 12 必須。

2 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 6 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 7 ハイパースレッディングとも呼ばれる同時マルチスレッドを有効または無効にします。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

9 マシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれている必要があります。

10 CIDR には、**platform.ibmcloud.controlPlaneSubnets** および **platform.ibmcloud.computeSubnets** で定義されたサブネットが含まれている必要があります。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

13 既存のリソースグループの名前。インストーラーによってプロビジョニングされたすべてのクラスターリソースは、このリソースグループにデプロイされます。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

14 VPC のネットワーク制限に基づいて、必要に応じて代替サービスエンドポイントを指定します。これにより、サービスのデフォルトのパブリックエンドポイントがオーバーライドされます。

15 既存の Virtual Private Cloud (VPC) を含むリソースグループの名前を指定します。既存の VPC およびサブネットはこのリソースグループにある必要があります。クラスターはこの VPC にインストールされます。

16 既存 VPC の名前を指定します。

17 コントロールプレーンマシンをデプロイする既存のサブネット名を指定します。サブネットは、指定した VPC に属している必要があります。リージョン内の各アベイラビリティーゾーンのサブネットを指定します。

18 コンピュートマシンをデプロイする既存のサブネット名を指定します。サブネットは、指定した VPC に属している必要があります。リージョン内の各アベイラビリティーゾーンのサブネットを指定します。

19 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: registry.example.com または registry.example.com:5000**<credentials>** について、ミラーレジストリーの base64 でエンコード

されたユーザー名およびパスワードを指定します。

20 FIPS モードを有効または無効にします。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

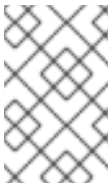
重要

プロセス暗号化ライブラリーでの FIPS 検証済みまたはモジュールの使用は、**x86_64** アーキテクチャーでの OpenShift Container Platform デプロイメントでのみサポートされます。

21 オプション: クラスター内のマシンにアクセスするのに使用する **sshKey** 値をオプションで指定できます。

22 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

23 レジストリーをミラーリングしたときに作成された **imageContentSourcePolicy.yaml** ファイルの **metadata.name: release-0** セクションからこれらの値を指定します。



注記

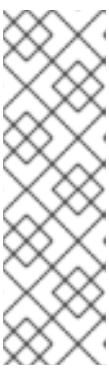
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

10.9.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

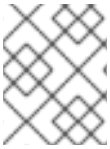
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

10.9.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
   PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

10.9.9. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
- architecture: amd64
  hyperthreading: Enabled
```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ openshift-install create manifests --dir <installation_directory>
```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
```

```
--included \ 1
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
--to=<path_to_directory_for_credentials_requests> 3
```

1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

2 **install-config.yaml** ファイルの場所を指定します。

3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

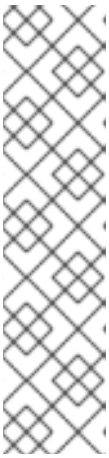
```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
  --credentials-requests-dir=<path_to_credential_requests_directory> \ 1
  --name=<cluster_name> \ 2
```

```
--output-dir=<installation_directory> \ 3
--resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

```
$ grep resourceGroupName <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

10.9.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
  Red Hat Enterprise Linux CoreOS (RHCOS) イメージがローカルで利用可能な場合、インストールプログラムを実行するホストはインターネットアクセスを必要としません。

- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

1. 次のコマンドを実行して、**OPENSHIFT_INSTALL_OS_IMAGE_OVERRIDE** 変数をエクスポートし、Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所を指定します。

```
$ export OPENSHIFT_INSTALL_OS_IMAGE_OVERRIDE="<path_to_image>/rhcos-<image_version>-ibmcloud.x86_64.qcow2.gz"
```

2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

10.9.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

10.9.12. インストール後の設定

以下の手順を実行して、クラスターの設定を完了します。

10.9.12.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

10.9.12.2. クラスターへのポリシーリソースのインストール

oc-mirror OpenShift CLI (oc) プラグインを使用して OpenShift Container Platform コンテンツをミラーリングすると、**catalogSource-certified-operator-index.yaml** および **imageContentSourcePolicy.yaml** を含むリソースが作成されます。

- **ImageContentSourcePolicy** リソースは、ミラーレジストリーをソースレジストリーに関連付け、イメージプル要求をオンラインレジストリーからミラーレジストリーにリダイレクトします。
- **CatalogSource** リソースは、Operator Lifecycle Manager (OLM) によって使用され、ミラーレジストリーで使用可能な Operator に関する情報を取得します。これにより、ユーザーは Operator を検出してインストールできます。

クラスターをインストールしたら、これらのリソースをクラスターにインストールする必要があります。

前提条件

- 非接続環境で、イメージセットをレジストリーミラーにミラーリングしました。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift CLI にログインします。
2. results ディレクトリーからクラスターに YAML ファイルを適用します。

```
$ oc apply -f ./oc-mirror-workspace/results-<id>/
```

検証

1. **ImageContentSourcePolicy** リソースが正常にインストールされたことを確認します。

```
$ oc get imagecontentsourcepolicy
```

2. **CatalogSource** リソースが正常にインストールされたことを確認します。

```
$ oc get catalogsource --all-namespaces
```

10.9.13. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

10.9.14. 次のステップ

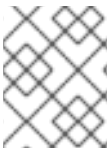
- クラスターをカスタマイズします。
- オプション: リモートヘルスレポートのオプトアウト

10.10. IBM CLOUD のインストール設定パラメーター

OpenShift Container Platform クラスターを IBM Cloud® にデプロイする前に、クラスターとそれをホストするプラットフォームをカスタマイズするパラメーターを指定します。**install-config.yaml** ファイルを作成するときは、コマンドラインを使用して必要なパラメーターの値を指定します。その後、**install-config.yaml** ファイルを変更して、クラスターをさらにカスタマイズできます。

10.10.1. IBM Cloud で使用可能なインストール設定パラメーター

次の表では、インストールプロセスの一部として設定できる、必須、オプション、および IBM Cloud 固有のインストール設定パラメーターを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

10.10.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表10.19 必須パラメーター

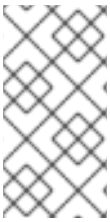
パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object

パラメーター	説明	値
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

10.10.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。



注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表10.20 ネットワークパラメーター

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。クラスターを既存の Virtual Private Cloud (VPC) にデプロイする場合、CIDR には **platform.ibmcloud.controlPlaneSubnets** および **platform.ibmcloud.computeSubnets** で定義されたサブネットが含まれている必要があります。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

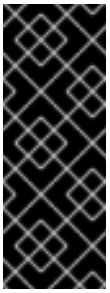
10.10.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表10.21 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String

パラメーター	説明	値
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String

パラメーター	説明	値
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]
	FIPS モードを有効または無効にしま	**false** または **true**

パラメーター	説明	値
fips:	す。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。 **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。 **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	

パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

10.10.1.4. 追加の IBM Cloud 設定パラメーター

追加の IBM Cloud® 設定パラメーターについて、以下の表で説明します。

表10.22 追加の IBM Cloud(R) パラメーター

パラメーター	説明	値
controlPlane: platform: ibmcloud: bootVolume: encryptionKey:	IBM® Key Protect for IBM Cloud® (Key Protect) のルート鍵。コントロールプレーンマシンのルート (ブート) ボリュームの暗号化にのみ使用します。	ルート鍵の Cloud Resource Name (CRN)。 CRN は引用符 ("") で囲む必要があります。

パラメーター	説明	値
compute: platform: ibmcloud: bootVolume: encryptionKey:	Key Protect ルート鍵。コンピュートマシンのみのルート (ブート) ボリュームの暗号化にのみ使用します。	ルート鍵の CRN。 CRN は引用符 ("") で囲む必要があります。

パラメーター	説明	値
platform: ibmcloud: defaultMachinePlatform: bootvolume: encryptionKey:	Key Protect ルート鍵。クラスターのマシンすべてのルート (ブート) ボリュームの暗号化に使用します。 デフォルトのマシン設定の一部として指定されると、すべてのマネージドストレージクラスがこの鍵で更新されます。そのため、インストール後にプロビジョニングされるデータボリュームもこの鍵を使用して暗号化されます。	ルート鍵の CRN。 CRN は引用符 ("") で囲む必要があります。
platform: ibmcloud: resourceGroupName:	既存のリソースグループの名前。デフォルトでは、インストーラーによってプロビジョニングされた VPC およびクラスターリソースは、このリソースグループに配置されます。指定しない場合、インストールプログラムはクラスターのリソースグループを作成します。クラスターを既存の VPC にデプロイする場合、インストーラーによってプロビジョニングされたクラスターリソースは、このリソースグループに配置されます。指定しない場合、インストールプログラムはクラスターのリソースグループを作成します。プロビジョニングした VPC リソースは、**networkResourceGroupName** パラメーターを使用して指定したリソースグループに存在する必要があります。いずれの場合も、クラスターコンポーネントはリソースグループ内のすべてのリソースの所有権を引き受けるため、このリソースグループは単一のクラスターインストールのみに使用する必要があります。[1]	文字列 (例: **existing_resource_group**)。

パラメーター	説明	値
platform: ibmcloud: serviceEndpoints: - name: url:	サービスエンドポイント名と URI のリスト。 デフォルトでは、インストールプログラムとクラスターコンポーネントはパブリックサービスエンドポイントを使用して、必要な IBM Cloud® サービスにアクセスします。 ネットワーク制限によってパブリックサービスエンドポイントへのアクセスが制限されている場合は、代替サービスエンドポイントを指定してデフォルトの動作をオーバーライドできます。 次の各サービスに、代替サービスエンドポイントを 1 つだけ指定できます。 • Cloud Object Storage • DNS Services • Global Search • Global Tagging • Identity Services • Key Protect • Resource Controller • Resource Manager • VPC	有効なサービスエンドポイント名と完全修飾 URI。 有効な名前は次のとおりです。 • **COS** • **DNSServices** • **GlobalServices** • **GlobalTagging** • **IAM** • **KeyProtect** • **ResourceController** • **ResourceManager** • **VPC**
platform: ibmcloud: networkResourceGroupName:	既存のリソースグループの名前。このリソースには、クラスターがデプロイされる既存の VPC とサブネットが含まれています。このパラメーターは、プロビジョニングした VPC にクラスターをデプロイする際に必要です。	文字列 (例: **existing_network_resource_group**)。

パラメーター	説明	値
platform: ibmcloud: dedicatedHosts: profile:	作成する新しい専用ホスト。**platform.ibmcloud.dedicatedHosts.name** に値を指定する場合、このパラメーターは必須ではありません。	**cx2-host-152x304** などの有効な IBM Cloud® 専用ホストプロファイル。 [2]
platform: ibmcloud: dedicatedHosts: name:	既存の専用ホスト。**platform.ibmcloud.dedicatedHosts.profile** に値を指定する場合、このパラメーターは必須ではありません。	文字列、たとえば **my-dedicated-host-name**。
platform: ibmcloud: type:	すべての IBM Cloud® マシンのインスタンスタイプ。	**bx2-8x32** などの有効な IBM Cloud® インスタンスタイプ。 [2]

パラメーター	説明	値
platform: ibmcloud: vpcName:	クラスターをデプロイする既存 VPC の名前。	文字列。
platform: ibmcloud: controlPlaneSubnets:	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。各アベイラビリティーゾーンのサブネットを指定します。	文字列配列
platform: ibmcloud: computeSubnets:	コンピュートマシンをデプロイする VPC の既存サブネットの名前。各アベイラビリティーゾーンのサブネットを指定します。サブネット ID はサポートされていません。	文字列配列

1. 既存のリソースグループを定義するか、インストーラーが作成するかによって、クラスターがアンインストールされたときにリソースグループがどのように扱われるかが決まります。リ

ソースグループを定義すると、インストーラーはインストーラーがプロビジョニングしたすべてのリソースを削除しますが、リソースグループはそのままにします。インストールの一部としてリソースグループが作成された場合、インストーラーは、インストーラーがプロビジョニングしたすべてのリソースとリソースグループを削除します。

2. 自身のニーズに最適なプロファイルを判別するには、IBM® ドキュメントの Instance Profiles を参照してください。

10.11. IBM CLOUD でのクラスターのアンインストール

IBM Cloud® にデプロイしたクラスターを削除できます。

10.11.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。
- **ccoctl** バイナリーを設定している。
- IBM Cloud® CLI をインストールし、VPC インフラストラクチャーサービスプラグインをインストールまたは更新している。詳細は、IBM Cloud® CLI ドキュメント の "Prerequisites" を参照してください。

手順

1. 次の条件が満たされている場合、この手順が必要です。
   - インストーラーは、インストールプロセスの一環としてリソースグループを作成しました。
   - クラスターがデプロイされた後、ユーザーまたはお使いのアプリケーションの 1 つが永続ボリューム要求 (PVC) を作成しました。

   この場合、クラスターをアンインストールするときに PVC が削除されないため、リソースグループが正常に削除されない可能性があります。失敗を防ぐには、以下を行います。

   a. CLI を使用して IBM Cloud® にログインします。

   b. PVC をリスト表示するには、次のコマンドを実行します。

   ```
   $ ibmcloud is volumes --resource-group-name <infrastructure_id>
   ```

ボリュームのリストの詳細については、IBM Cloud® CLI のドキュメント を参照してください。

c. PVC を削除するには、次のコマンドを実行します。

```
$ ibmcloud is volume-delete --force <volume_id>
```

ボリュームの削除の詳細は、IBM Cloud® CLI のドキュメント を参照してください。

2. インストールプロセスの一環として作成された API キーをエクスポートします。

```
$ export IC_API_KEY=<api_key>
```

注記

変数名は指定どおりに設定する必要があります。インストールプログラムは、クラスターのインストール時に作成されたサービス ID を削除するために、変数名が存在することを想定しています。

3. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

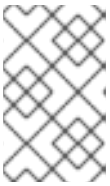
2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

4. クラスター用に作成された手動の CCO クレデンシャルを削除します。

```
$ ccoctl ibmcloud delete-service-id \
    --credentials-requests-dir <path_to_credential_requests_directory> \
    --name <cluster_name>
```



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

5. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第11章 NUTANIX へのインストール

11.1. NUTANIX へのインストールの準備

OpenShift Container Platform クラスターをインストールする前に、Nutanix 環境が以下の要件を満たしていることを確認してください。

11.1.1. Nutanix バージョン要件

以下の要件を満たす Nutanix 環境に OpenShift Container Platform クラスターをインストールする必要があります。

表11.1 Nutanix 仮想環境のバージョン要件

コンポーネント	必須バージョン
Nutanix AOS	6.5.2.7 以降
Prism Central	pc.2022.6 以降

11.1.2. 環境要件

OpenShift Container Platform クラスターをインストールする前に、以下の Nutanix AOS 環境要件を確認してください。

11.1.2.1. 必要なアカウント特権

インストールプログラムには、クラスターをデプロイし、その日次の操作を維持するために必要な権限のある Nutanix アカウントへのアクセスが必要です。以下のオプションを使用できます。

- 管理者権限を持つローカル Prism Central ユーザーアカウントを使用できます。ローカルアカウントを使用することは、必要な権限を持つアカウントへのアクセスを許可する最も簡単な方法です。
- 組織のセキュリティーポリシーにより、より制限の厳しい権限セットを使用する必要がある場合は、次の表にリストされている権限を使用して、Prism Central でカスタムの Cloud Native ロールを作成します。その後、Prism Central 認証ディレクトリーのメンバーであるユーザーアカウントにロールを割り当てることができます。

このユーザーアカウントを管理するときは、次の点を考慮してください。

- エンティティーをロールに割り当てるときは、ユーザーが仮想マシンのデプロイに必要な Prism Element とサブネットのみにアクセスできるようにしてください。
- ユーザーが、仮想マシンを割り当てる必要があるプロジェクトのメンバーであることを確認してください。

詳細は、Custom Cloud Native ロール の作成、ロールの割り当て、プロジェクトへのユーザーの追加に関する Nutanix ドキュメントを参照してください。

> **例11.1 Custom Cloud Native ロールの作成に必要な権限**

Nutanix オブジェクト	必要になる場合	Nutanix API で必要な権限	説明
Categories	Always	**Create_Category_Mapping** **Create_Or_Update_Name_Category** **Create_Or_Update_Value_Category** **Delete_Category_Mapping** **Delete_Name_Category** **Delete_Value_Category** **View_Category_Mapping** **View_Name_Category** **View_Value_Category**	OpenShift Container Platform マシンに割り当てられたカテゴリーを作成、読み取り、削除します。
Images	Always	**Create_Image** **Delete_Image** **View_Image**	OpenShift Container Platform マシンに使用されるオペレーティングシステムイメージを作成、読み取り、削除します。
仮想マシン	Always	**Create_Virtual_Machine** **Delete_Virtual_Machine** **View_Virtual_Machine**	OpenShift Container Platform マシンを作成、読み取り、削除します。
クラスター	Always	**View_Cluster**	OpenShift Container Platform マシンをホストする Prism Element クラスターを表示します。
サブネット	Always	**View_Subnet**	OpenShift Container Platform マシンをホストするサブネットを表示します。

Nutanix オブジェクト	必要になる場合	Nutanix API で必要な権限	説明
プロジェクト	プロジェクトをコンピューティングマシン、コントロールプレーンマシン、またはすべてのマシンに関連付ける場合。	**View_Project**	Prism Central で定義されたプロジェクトを表示し、プロジェクトを OpenShift Container Platform マシンに割り当てることができるようにします。

11.1.2.2. クラスターの制限

利用可能なリソースはクラスターによって異なります。Nutanix 環境内で可能なクラスターの数は、主に、使用可能なストレージスペースと、クラスターが作成するリソースに関連する制限、および IP アドレスやネットワークなど、クラスターをデプロイするために必要なリソースによって制限されます。

11.1.2.3. クラスターリソース

標準クラスターを使用するには、最低 800 GB のストレージが必要です。

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは Nutanix インスタンスに複数のリソースを作成できる必要があります。これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはインストールプロセスの一部として破棄されます。

標準的な OpenShift Container Platform インストールでは、以下のリソースを作成します。

- 1 ラベル
- 仮想マシン:
  - 1 ディスクイメージ
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

11.1.2.4. ネットワーク要件

ネットワークには AHV IP アドレス管理 (IPAM) または動的ホスト設定プロトコル (DHCP) のいずれかを使用し、クラスターマシンに永続的な IP アドレスを提供するように設定されていることを確認する必要があります。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作します。

- IP アドレス
- DNS レコード

注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、NTP サーバーは、非同期サーバークロックに通常関連するエラーを防ぎます。

11.1.2.4.1. 必要な IP アドレス

インストーラーによってプロビジョニングされたインストールには、2 つの静的仮想 IP (VIP) アドレスが必要です。

- API の VIP アドレスが必要です。このアドレスは、クラスター API にアクセスするために使用されます。
- Ingress 用の VIP アドレスが必要です。このアドレスは、クラスターの Ingress トラフィックに使用されます。

これらの IP アドレスは、OpenShift Container Platform クラスターをインストールするときに指定します。

11.1.2.4.2. DNS レコード

OpenShift Container Platform クラスターをホストする Nutanix インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。

独自の DNS または DHCP サーバーを使用する場合は、ブートストラップ、コントロールプレーン、コンピュートノードなどの各ノードのレコードも作成する必要があります。

完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表11.2 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

11.1.3. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報を Kubernetes カスタムリソース定義 (CRD) として管理します。Nutanix にクラスターをインストールするには、インストールプロセスの一部として CCO を **manual** モードに設定する必要があります。

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

関連情報

- 手動で維持された認証情報でクラスターを更新する準備

11.2. 複数の PRISM ELEMENT を使用したフォールトトレラントなデプロイメント

デフォルトでは、インストールプログラムは、コントロールプレーンとコンピュートマシンを単一の Nutanix Prism Element (クラスター) にインストールします。OpenShift Container Platform クラスターのフォールトトレランスを向上させるために、これらのマシンを複数の Nutanix クラスターに分散するように指定できます。これを行うには、障害ドメインを設定します。

障害ドメインは、インストール中およびインストール後に OpenShift Container Platform マシンプールで使用できる追加の Prism Element インスタンスを表します。

11.2.1. 障害ドメインの要件

障害ドメインの使用を計画する場合は、次の要件を考慮してください。

- すべての Nutanix Prism Element インスタンスは、同一の Prism Central インスタンスによって管理する必要があります。複数の Prism Central インスタンスで構成されるデプロイメントはサポートされていません。
- Prism Element クラスターを構成するマシンは、障害ドメインが相互に通信できるように、同じイーサネットネットワーク上に存在する必要があります。
- OpenShift Container Platform クラスターで障害ドメインとして使用する各 Prism Element には、サブネットが必要です。これらのサブネットを定義するときは、共通の IP アドレス接頭辞 (CIDR) を指定する必要があります。また、OpenShift Container Platform クラスターが使用する仮想 IP アドレスをサブネットに含める必要があります。

11.2.2. インストール方法と障害ドメインの設定

OpenShift Container Platform のインストール方法によって、障害ドメインをいつどのように設定するかが決まります。

- installer-provisioned infrastructure を使用してデプロイする場合は、クラスターをデプロイする前に、インストール設定ファイルで障害ドメインを設定できます。詳細は、障害ドメインの設定 を参照してください。
  クラスターのデプロイ後に障害ドメインを設定することもできます。インストール後の障害ドメインの設定の詳細は、既存の Nutanix クラスターへの障害ドメインの追加 を参照してください。
- お客様が管理するインフラストラクチャー (user-provisioned infrastructure) を使用してデプロイする場合、追加の設定は必要ありません。クラスターをデプロイした後、コントロールプレーンとコンピュートマシンを障害ドメイン全体に手動で分散できます。

11.3. クラスターの NUTANIX へのインストール

OpenShift Container Platform バージョン 4.16 では、次のいずれかの方法を選択して、Nutanix インスタンスにクラスターをインストールできます。

インストーラーによってプロビジョニングされたインフラストラクチャーの使用: インストーラーによってプロビジョニングされたインフラストラクチャーを使用するには、次のセクションの手順を使用します。インストーラーによってプロビジョニングされたインフラストラクチャーは、接続されたネッ

トワーク環境または切断されたネットワーク環境でのインストールに最適です。インストーラーによってプロビジョニングされたインフラストラクチャーには、クラスターの基礎となるインフラストラクチャーをプロビジョニングするインストールプログラムが含まれています。

Assisted Installer の使用: Assisted Installer は console.redhat.com でホストされます。アシステッドインストーラーは、切断された環境では使用できません。アシステッドインストーラーはクラスターの基礎となるインフラストラクチャーをプロビジョニングしないため、アシステッドインストーラーを実行する前にインフラストラクチャーをプロビジョニングする必要があります。Assisted Installer を使用してインストールすると、Nutanix との統合も提供され、自動スケーリングが可能になります。詳細は、自動インストーラーを使用したオンプレミスクラスターのインストール を参照してください。

ユーザーによってプロビジョニングされたインフラストラクチャーの使用: 任意のプラットフォームへのクラスターのインストール ドキュメントに概説されている関連手順を完了します。

11.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- インストールプログラムで、Prism Central および Prism Element のポート 9440 にアクセスできる。ポート 9440 にアクセスできることを確認している。
- ファイアウォールを使用している場合は、次の前提条件を満たしています。
  - ポート 9440 にアクセスできることを確認している。コントロールプレーンノードがポート 9440 で Prism Central および Prism Element にアクセスできる (インストールが成功するために必要)。
  - OpenShift Container Platform が必要とするサイトへの アクセスを許可する ようにファイアウォールを設定している。これには、Telemetry の使用が含まれます。
- Nutanix 環境でデフォルトの自己署名 SSL 証明書を使用している場合は、CA によって署名された証明書に置き換える。インストールプログラムには、Prism Central API にアクセスするための有効な CA 署名付き証明書が必要です。自己署名証明書の置き換えに関する詳細は、Nutanix AOS Security Guide を参照してください。
  Nutanix 環境で内部 CA を使用して証明書を発行する場合は、インストールプロセスの一部としてクラスター全体のプロキシーを設定する必要があります。詳細は、カスタム PKI の設定 を参照してください。

重要

2048 ビット証明書を使用します。Prism Central 2022.x で 4096 ビット証明書を使用すると、インストールに失敗します。

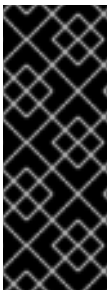
11.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。

- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

11.3.3. Prism Central のインターネットアクセス

Prism Central では、クラスターのインストールに必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得するためにインターネットアクセスが必要です。Nutanix の RHCOS イメージは、**rhcos.mirror.openshift.com** で入手できます。

11.3.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

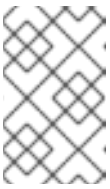
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

11.3.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

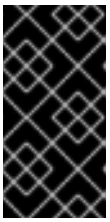
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

11.3.6. Nutanix root CA 証明書をシステム信頼に追加する

インストールプログラムは Prism Central API へのアクセスを必要とするため、OpenShift Container Platform クラスターをインストールする前に、Nutanix の信頼された root CA 証明書をシステム信頼に追加する必要があります。

手順

1. Prism Central Web コンソールから、Nutanix root CA 証明書をダウンロードします。
2. Nutanix root CA 証明書を含む圧縮ファイルを抽出します。
3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

11.3.7. インストール設定ファイルの作成

Nutanix にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- Nutanix のネットワーク要件を満たしていることが確認されました。詳細については、Nutanix へのインストールの準備を参照してください。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールア

セットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットとするプラットフォームとして **nutanix** を選択します。

iii. Prism Central のドメイン名または IP アドレスを入力します。

iv. Prism Central へのログインに使用するポートを入力します。

v. Prism Central へのログインに使用する認証情報を入力します。
インストールプログラムが Prism Central に接続します。

vi. OpenShift Container Platform クラスターを管理する Prism Element を選択します。

vii. 使用するネットワークサブネットを選択します。

viii. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

ix. クラスター Ingress に設定した仮想 IP アドレスを入力します。

x. ベースドメインを入力します。このベースドメインは、DNS レコードで設定したものと同じである必要があります。

xi. クラスターの記述名を入力します。
入力するクラスター名は、DNS レコードの設定時に指定したクラスター名と一致する必要があります。

2. オプション: **install.config.yaml** ファイル内の 1 つ以上のデフォルト設定パラメーターを更新して、インストールをカスタマイズします。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

注記

3 ノードクラスターをインストールする場合は、必ず **compute.replicas** パラメーターを **0** に設定してください。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細は、「Nutanix への 3 ノードクラスターのインストール」を参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- Nutanix のインストール設定パラメーター

11.3.7.1. Nutanix 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    nutanix: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMiB: 8196
      osDisk:
        diskSizeGiB: 120
      categories: (5)
      - key: <category_key_name>
        value: <category_value>
controlPlane: (6)
  hyperthreading: Enabled (7)
  name: master
  replicas: 3
  platform:
    nutanix: (8)
      cpus: 4
      coresPerSocket: 2
      memoryMiB: 16384
      osDisk:
        diskSizeGiB: 120
      categories: (9)
      - key: <category_key_name>
        value: <category_value>
metadata:
  creationTimestamp: null
  name: test-cluster (10)
```

```
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (11)
  serviceNetwork:
    - 172.30.0.0/16
platform:
  nutanix:
    apiVIPs:
      - 10.40.142.7 (12)
    defaultMachinePlatform:
      bootType: Legacy
      categories: (13)
      - key: <category_key_name>
        value: <category_value>
      project: (14)
        type: name
        name: <project_name>
    ingressVIPs:
      - 10.40.142.8 (15)
    prismCentral:
      endpoint:
        address: your.prismcentral.domainname (16)
        port: 9440 (17)
      password: <password> (18)
      username: <username> (19)
    prismElements:
    - endpoint:
        address: your.prismelement.domainname
        port: 9440
      uuid: 0005b0f1-8f43-a0f2-02b7-3cecef193712
    subnetUUIDs:
    - c7938dc6-7659-453e-a688-e26020c68e43
    clusterOSImage: http://example.com/images/rhcos-47.83.202103221318-0-nutanix.x86_64.qcow2
(20)
credentialsMode: Manual
publish: External
pullSecret: '{"auths": ...}' (21)
fips: false (22)
sshKey: ssh-ed25519 AAAA... (23)
```

(1) (10) (12) (15) (16) (17) (18) (19) (21) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(2) (6) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 8 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

5 9 13 オプション: プリズムカテゴリーキーとプリズムカテゴリー値のペアを 1 つ以上指定します。これらのカテゴリーのキーと値のペアは、Prism Central に存在する必要があります。コンピューティングマシン、コントロールプレーンマシン、またはすべてのマシンに個別のカテゴリーを指定できます。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

14 オプション: VM が関連付けられているプロジェクトを指定します。プロジェクトタイプの **name** または **uuid** を指定してから、対応する UUID またはプロジェクト名を指定します。プロジェクトは、コンピューティングマシン、コントロールプレーンマシン、またはすべてのマシンに関連付けることができます。

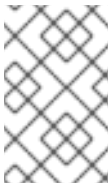
20 オプション: デフォルトでは、インストールプログラムは Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードしてインストールします。Prism Central がインターネットにアクセスできない場合は、任意の HTTP サーバーで RHCOS イメージをホストし、インストールプログラムがそのイメージを指すようにすることで、デフォルトの動作をオーバーライドできます。

22 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

23 オプション: クラスター内のマシンにアクセスするのに使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

11.3.7.2. 障害ドメインの設定

障害ドメインは、コントロールプレーンとコンピュートマシンを複数の Nutanix Prism Element (クラスター) に分散することで、OpenShift Container Platform クラスターのフォールトトレランスを向上させます。

ヒント

高可用性を確保するには、3 つの障害ドメインを設定することを推奨します。

前提条件

- インストール設定ファイル (**install-config.yaml**) がある。

手順

1. **install-config.yaml** ファイルを編集し、次のスタンザを追加して最初の障害ドメインを設定します。

```
apiVersion: v1
baseDomain: example.com
compute:
# ...
platform:
  nutanix:
    failureDomains:
    - name: <failure_domain_name>
      prismElement:
        name: <prism_element_name>
        uuid: <prism_element_uuid>
      subnetUUIDs:
      - <network_uuid>
# ...
```

ここでは、以下のようになります。

<failure_domain_name>

障害ドメインの一意の名前を指定します。名前は 64 文字以下に制限されており、小文字、数字、ダッシュ (**-**) を含めることができます。ダッシュを名前の先頭または末尾に含めることはできません。

<prism_element_name>

オプション: Prism Element の名前を指定します。

<prism_element_uuid>

Prism Element の UUID を指定します。

<network_uuid>

Prism Element サブネットオブジェクトの UUID を指定します。サブネットの IP アドレス接頭辞 (CIDR) には、OpenShift Container Platform クラスターが使用する仮想 IP アドレスを含める必要があります。OpenShift Container Platform クラスター内の障害ドメイン (Prism Element) ごとに 1 つのサブネットのみがサポートされます。

2. 必要に応じて、追加の障害ドメインを設定します。

3. コントロールプレーンとコンピュートマシンを障害ドメイン全体に分散するには、次のいずれかを実行します。

- コンピューティングプレーンとコントロールプレーンのマシンが同じ障害ドメインセットを共有できる場合は、クラスターのデフォルトのマシン設定の下に障害ドメイン名を追加します。

  障害ドメインセットを共有するコントロールプレーンとコンピュートマシンの例

```
apiVersion: v1
baseDomain: example.com
compute:
# ...
platform:
  nutanix:
    defaultMachinePlatform:
      failureDomains:
        - failure-domain-1
        - failure-domain-2
        - failure-domain-3
# ...
```

- コンピュートマシンとコントロールプレーンマシンが別々の障害ドメインを使用する必要がある場合は、それぞれのマシンプールの下に障害ドメイン名を追加します。

  別々の障害ドメインを使用するコントロールプレーンとコンピュートマシンの例

```
apiVersion: v1
baseDomain: example.com
compute:
# ...
controlPlane:
  platform:
    nutanix:
      failureDomains:
        - failure-domain-1
        - failure-domain-2
        - failure-domain-3
# ...
compute:
  platform:
    nutanix:
      failureDomains:
        - failure-domain-1
        - failure-domain-2
# ...
```

4. ファイルを保存します。

11.3.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** で

クライアントの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

11.3.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

11.3.9. Nutanix の IAM の設定

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムが手動モード用に CCO を設定する間、ID およびアクセス管理シークレットを指定する必要があります。

前提条件

- **ccoctl** バイナリーを設定している。
- **install-config.yaml** ファイルがある。

手順

1. 認証情報データを含む YAML ファイルを次の形式で作成します。

認証情報データの形式

```
credentials:
- type: basic_auth (1)
  data:
    prismCentral: (2)
      username: <username_for_prism_central>
      password: <password_for_prism_central>
    prismElements: (3)
    - name: <name_of_prism_element>
      username: <username_for_prism_element>
      password: <password_for_prism_element>
```

(1) 認証タイプを指定します。Basic 認証のみがサポートされています。

2 Prism Central の認証情報を指定します。

3 オプション: Prism Element 認証情報を指定します。

2. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

2 **install-config.yaml** ファイルの場所を指定します。

3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  annotations:
    include.release.openshift.io/self-managed-high-availability: "true"
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-machine-api-nutanix
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: NutanixProviderSpec
  secretRef:
    name: nutanix-credentials
    namespace: openshift-machine-api
```

4. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl nutanix create-shared-secrets \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 1
  --output-dir=<ccoctl_output_dir> \ 2
```

```
--credentials-source-filepath=<path_to_credentials_file> 3
```

1. コンポーネント **CredentialsRequests** オブジェクトのファイルを含むディレクトリーへのパスを指定します。
2. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
3. オプション: 認証情報データ YAML ファイルを含むディレクトリーを指定します。デフォルトでは、**ccoctl** はこのファイルが **<home_directory>/.nutanix/credentials** にあると想定します。

5. **credentialsMode** パラメーターが **Manual** に設定されるように、**install-config.yaml** 設定ファイルを編集します。

   サンプル install-config.yaml 設定ファイル

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual 1
...
```

   1. この行を追加して、**credentialsMode** パラメーターを **Manual** に設定します。

6. 次のコマンドを実行して、インストールマニフェストを作成します。

```
$ openshift-install create manifests --dir <installation_directory> 1
```

   1. クラスターの **install-config.yaml** ファイルを含むディレクトリーへのパスを指定します。

7. 次のコマンドを実行して、生成された認証情報ファイルをターゲットマニフェストディレクトリーにコピーします。

```
$ cp <ccoctl_output_dir>/manifests/*credentials.yaml ./<installation_directory>/manifests
```

検証

- **manifests** ディレクトリーに適切なシークレットが存在することを確認します。

```
$ ls ./<installation_directory>/manifests
```

出力例

```
cluster-config.yaml
cluster-dns-02-config.yml
cluster-infrastructure-02-config.yml
cluster-ingress-02-config.yml
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-proxy-01-config.yaml
```

```
cluster-scheduler-02-config.yml
cvo-overrides.yaml
kube-cloud-config.yaml
kube-system-configmap-root-ca.yaml
machine-config-server-tls-secret.yaml
openshift-config-secret-pull-secret.yaml
openshift-cloud-controller-manager-nutanix-credentials-credentials.yaml
openshift-machine-api-nutanix-credentials-credentials.yaml
```

11.3.10. Nutanix CCM に必要な config map とシークレットリソースの追加

Nutanix にインストールするには、Nutanix Cloud Controller Manager (CCM) と統合するために追加の **ConfigMap** および **Secret** リソースが必要です。

前提条件

- インストールディレクトリー内に **manifests** ディレクトリーが作成されました。

手順

1. **manifests** ディレクトリーに移動します。

```
$ cd <path_to_installation_directory>/manifests
```

2. **openshift-cloud-controller-manager-cloud-config.yaml** という名前で **cloud-conf ConfigMap** ファイルを作成し、以下の情報を追加します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cloud-conf
  namespace: openshift-cloud-controller-manager
data:
  cloud.conf: "{
    \"prismCentral\": {
      \"address\": \"<prism_central_FQDN/IP>\", (1)
      \"port\": 9440,
       \"credentialRef\": {
          \"kind\": \"Secret\",
          \"name\": \"nutanix-credentials\",
          \"namespace\": \"openshift-cloud-controller-manager\"
       }
    },
    \"topologyDiscovery\": {
       \"type\": \"Prism\",
       \"topologyCategories\": null
    },
    \"enableCustomLabeling\": true
  }"
```

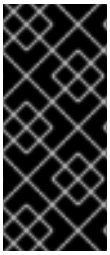
(1) Prism Central FQDN/IP を指定します。

3. ファイル **cluster-infrastructor-02-config.yml** が存在し、次の情報が含まれていることを確認します。

```
spec:
  cloudConfig:
    key: config
    name: cloud-provider-config
```

11.3.11. ユーザー管理ロードバランサーのサービス

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。



重要

ユーザー管理ロードバランサーの設定は、ベンダーのロードバランサーによって異なります。

このセクションの情報と例は、ガイドラインのみを目的としています。ベンダーのロードバランサーに関する詳細は、ベンダーのドキュメントを参照してください。

Red Hat は、ユーザー管理ロードバランサーに対して次のサービスをサポートしています。

- Ingress Controller
- OpenShift API
- OpenShift MachineConfig API

ユーザー管理ロードバランサーに対して、これらのサービスの 1 つを設定するか、すべてを設定するかを選択できます。一般的な設定オプションは、Ingress Controller サービスのみを設定することです。次の図は、各サービスの詳細を示しています。

図11.1 OpenShift Container Platform 環境で動作する Ingress Controller を示すネットワークワークフローの例

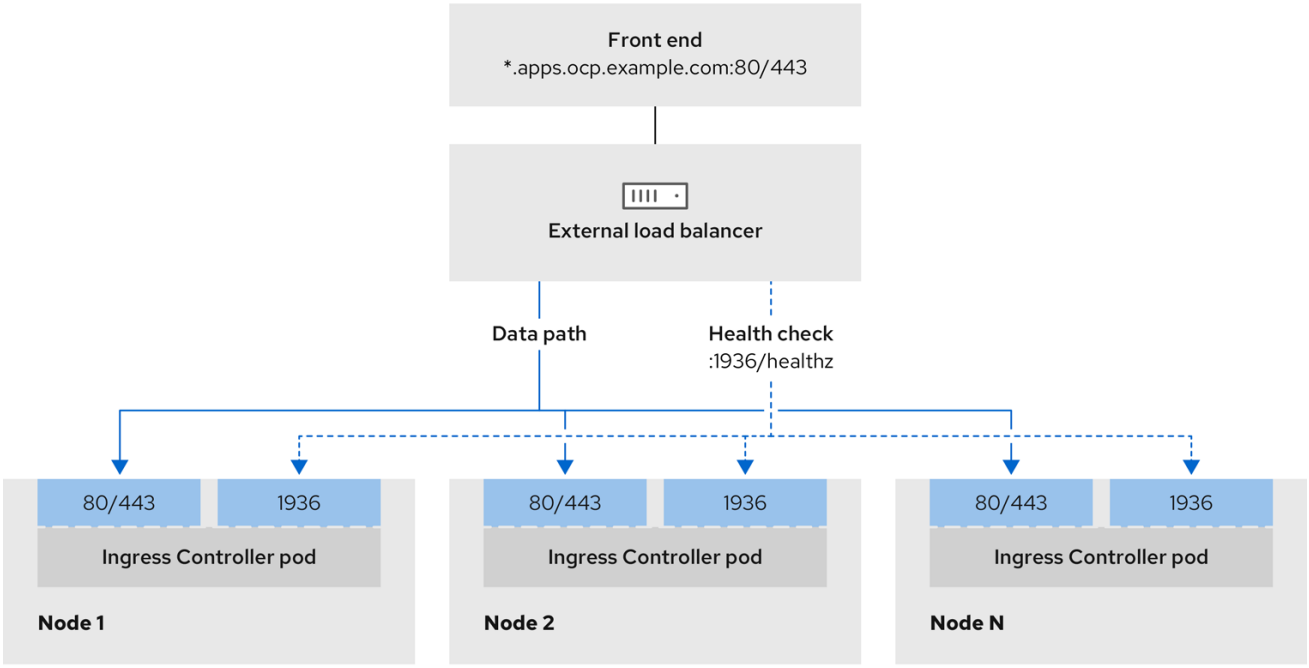


図11.2 OpenShift Container Platform 環境で動作する OpenShift API を示すネットワークワークフローの例

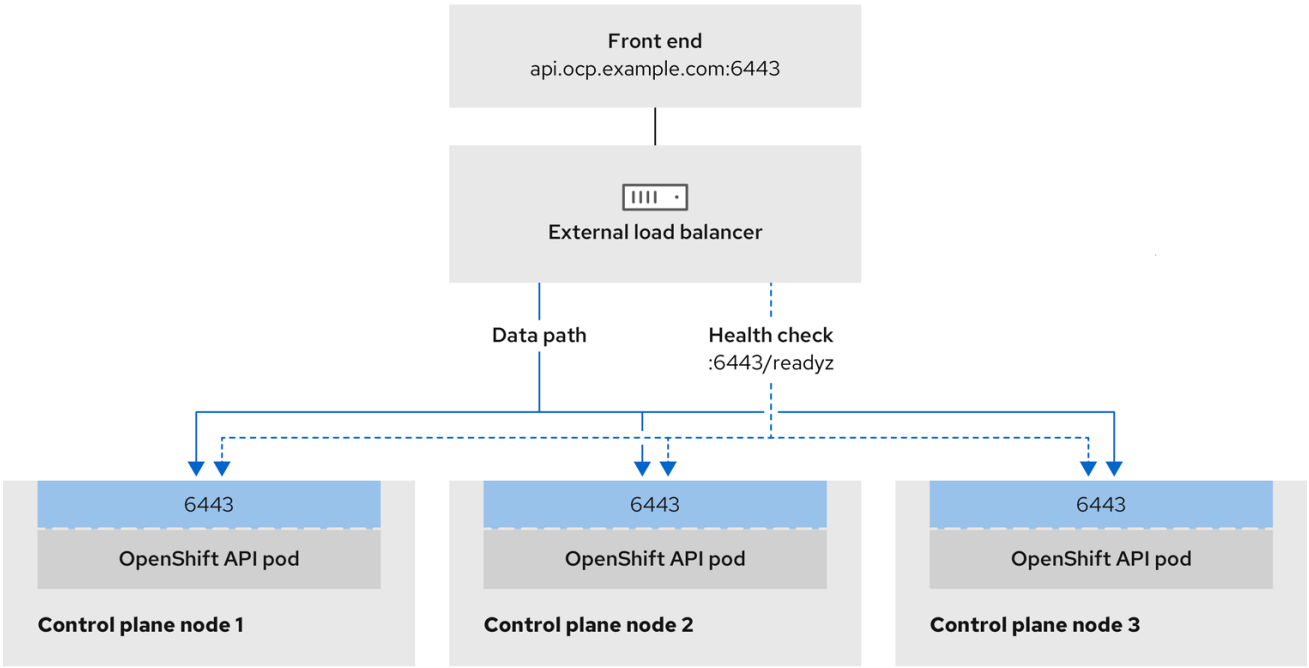
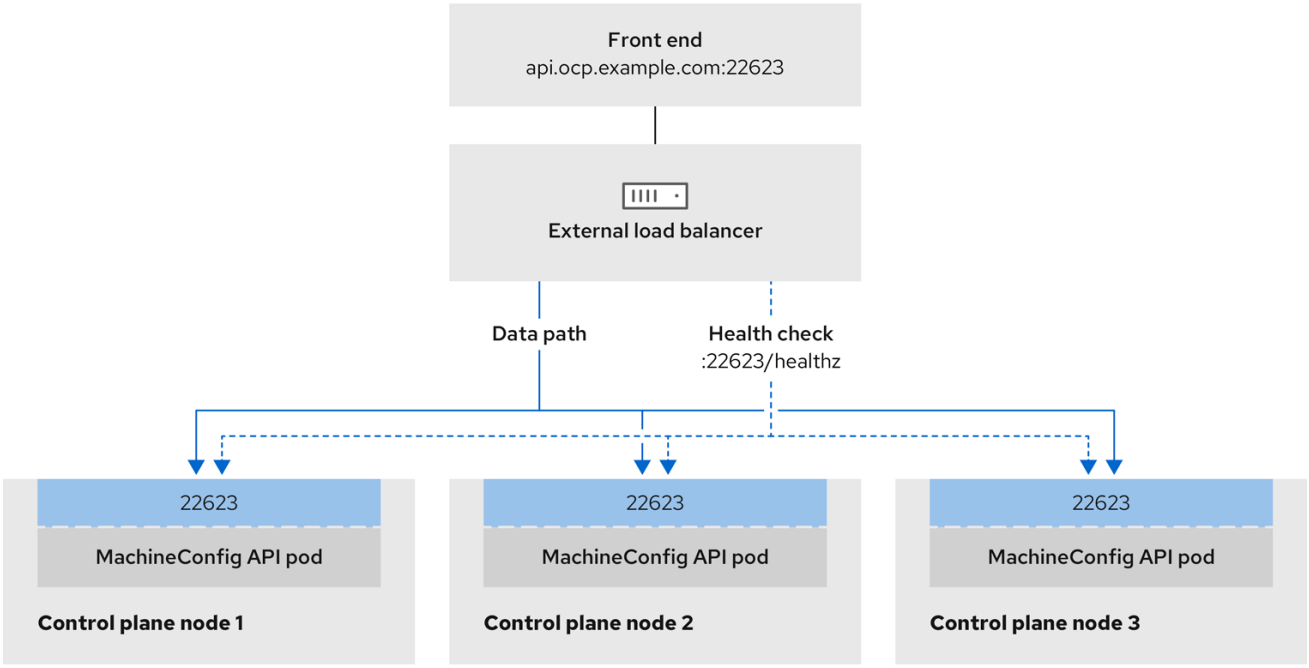


図11.3 OpenShift Container Platform 環境で動作する OpenShift MachineConfig API を示すネットワークワークフローの例



ユーザー管理ロードバランサーでは、次の設定オプションがサポートされています。

- ノードセレクターを使用して、Ingress Controller を特定のノードのセットにマッピングします。このセットの各ノードに静的 IP アドレスを割り当てるか、Dynamic Host Configuration Protocol (DHCP) から同じ IP アドレスを受け取るように各ノードを設定する必要があります。インフラストラクチャーノードは通常、このタイプの設定を受け取ります。
- サブネット上のすべての IP アドレスをターゲットにします。この設定では、ロードバランサーターゲットを再設定せずにネットワーク内でノードを作成および破棄できるため、メンテナンスオーバーヘッドを削減できます。**/27** や **/28** などの小規模なネットワーク上に設定されたマシンを使用して Ingress Pod をデプロイする場合、ロードバランサーのターゲットを簡素化できます。

  ヒント

  マシン config プールのリソースを確認することで、ネットワーク内に存在するすべての IP アドレスをリスト表示できます。

OpenShift Container Platform クラスターのユーザー管理ロードバランサーを設定する前に、以下の情報を考慮してください。

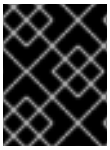
- フロントエンド IP アドレスの場合、フロントエンド IP アドレス、Ingress Controller のロードバランサー、および API ロードバランサーに同じ IP アドレスを使用できます。この機能については、ベンダーのドキュメントを確認してください。
- バックエンド IP アドレスの場合、ユーザー管理ロードバランサーの有効期間中に OpenShift Container Platform コントロールプレーンノードの IP アドレスが変更されないことを確認します。次のいずれかのアクションを実行すると、これを実現できます。
  - 各コントロールプレーンノードに静的 IP アドレスを割り当てます。

- ノードが DHCP リースを要求するたびに、DHCP から同じ IP アドレスを受信するように各ノードを設定します。ベンダーによっては、DHCP リースは IP 予約または静的 DHCP 割り当ての形式になる場合があります。

- Ingress Controller バックエンドサービスのユーザー管理ロードバランサーで Ingress Controller を実行する各ノードを手動で定義します。たとえば、Ingress Controller が未定義のノードに移動すると、接続が停止する可能性があります。

11.3.11.1. ユーザー管理ロードバランサーの設定

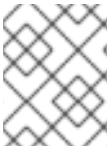
デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。



重要

ユーザー管理ロードバランサーを設定する前に、ユーザー管理ロードバランサーのサービス" セクションを必ずお読みください。

ユーザー管理ロードバランサー用に設定するサービスに適用される次の前提条件をお読みください。



注記

クラスター上で実行される MetalLB は、ユーザー管理ロードバランサーとして機能します。

OpenShift API の前提条件

- フロントエンド IP アドレスを定義している。

- TCP ポート 6443 および 22623 は、ロードバランサーのフロントエンド IP アドレスで公開されている。以下の項目を確認します。

  - ポート 6443 が OpenShift API サービスにアクセスできる。

  - ポート 22623 が Ignition 起動設定をノードに提供できる。

- フロントエンド IP アドレスとポート 6443 へは、OpenShift Container Platform クラスターの外部の場所にいるシステムのすべてのユーザーがアクセスできる。

- フロントエンド IP アドレスとポート 22623 は、OpenShift Container Platform ノードからのみ到達できる。

- ロードバランサーバックエンドは、ポート 6443 および 22623 の OpenShift Container Platform コントロールプレーンノードと通信できる。

Ingress Controller の前提条件

- フロントエンド IP アドレスを定義している。

- TCP ポート 443 および 80 はロードバランサーのフロントエンド IP アドレスで公開されている。

- フロントエンドの IP アドレス、ポート 80、ポート 443 へは、OpenShift Container Platform クラスターの外部の場所にあるシステムの全ユーザーがアクセスできる。

- フロントエンドの IP アドレス、ポート 80、ポート 443 は、OpenShift Container Platform クラスターで動作するすべてのノードから到達できる。
- ロードバランサーバックエンドは、ポート 80、443、および 1936 で Ingress Controller を実行する OpenShift Container Platform ノードと通信できる。

ヘルスチェック URL 仕様の前提条件

ほとんどのロードバランサーは、サービスが使用可能か使用不可かを判断するヘルスチェック URL を指定して設定できまうｓ．OpenShift Container Platform は、OpenShift API、Machine Configuration API、および Ingress Controller バックエンドサービスのこれらのヘルスチェックを提供します。

次の例は、前にリスト表示したバックエンドサービスのヘルスチェック仕様を示しています。

Kubernetes API ヘルスチェック仕様の例

```
Path: HTTPS:6443/readyz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Machine Config API ヘルスチェック仕様の例

```
Path: HTTPS:22623/healthz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Ingress Controller のヘルスチェック仕様の例

```
Path: HTTP:1936/healthz/ready
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 5
Interval: 10
```

手順

1. HAProxy Ingress Controller を設定して、ポート 6443、22623、443、および 80 でロードバランサーからクラスターへのアクセスを有効化できるようにします。必要に応じて、HAProxy 設定で単一のサブネットの IP アドレスまたは複数のサブネットの IP アドレスを指定できます。

   #### 1つのサブネットをリストした HAProxy 設定の例

   ```
   # ...
   listen my-cluster-api-6443
       bind 192.168.1.100:6443
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /readyz
   ```

```
  http-check expect status 200
    server my-cluster-master-2 192.168.1.101:6443 check inter 10s rise 2 fall 2
    server my-cluster-master-0 192.168.1.102:6443 check inter 10s rise 2 fall 2
    server my-cluster-master-1 192.168.1.103:6443 check inter 10s rise 2 fall 2

listen my-cluster-machine-config-api-22623
    bind 192.168.1.100:22623
    mode tcp
    balance roundrobin
  option httpchk
  http-check connect
  http-check send meth GET uri /healthz
  http-check expect status 200
    server my-cluster-master-2 192.168.1.101:22623 check inter 10s rise 2 fall 2
    server my-cluster-master-0 192.168.1.102:22623 check inter 10s rise 2 fall 2
    server my-cluster-master-1 192.168.1.103:22623 check inter 10s rise 2 fall 2

listen my-cluster-apps-443
    bind 192.168.1.100:443
    mode tcp
    balance roundrobin
  option httpchk
  http-check connect
  http-check send meth GET uri /healthz/ready
  http-check expect status 200
    server my-cluster-worker-0 192.168.1.111:443 check port 1936 inter 10s rise 2 fall 2
    server my-cluster-worker-1 192.168.1.112:443 check port 1936 inter 10s rise 2 fall 2
    server my-cluster-worker-2 192.168.1.113:443 check port 1936 inter 10s rise 2 fall 2

listen my-cluster-apps-80
   bind 192.168.1.100:80
   mode tcp
   balance roundrobin
  option httpchk
  http-check connect
  http-check send meth GET uri /healthz/ready
  http-check expect status 200
    server my-cluster-worker-0 192.168.1.111:80 check port 1936 inter 10s rise 2 fall 2
    server my-cluster-worker-1 192.168.1.112:80 check port 1936 inter 10s rise 2 fall 2
    server my-cluster-worker-2 192.168.1.113:80 check port 1936 inter 10s rise 2 fall 2
# ...
```

複数のサブネットをリストした HAProxy 設定の例

```
# ...
listen api-server-6443
    bind *:6443
    mode tcp
      server master-00 192.168.83.89:6443 check inter 1s
      server master-01 192.168.84.90:6443 check inter 1s
      server master-02 192.168.85.99:6443 check inter 1s
      server bootstrap 192.168.80.89:6443 check inter 1s

listen machine-config-server-22623
    bind *:22623
    mode tcp
```

```
    server master-00 192.168.83.89:22623 check inter 1s
    server master-01 192.168.84.90:22623 check inter 1s
    server master-02 192.168.85.99:22623 check inter 1s
    server bootstrap 192.168.80.89:22623 check inter 1s

listen ingress-router-80
  bind *:80
  mode tcp
  balance source
    server worker-00 192.168.83.100:80 check inter 1s
    server worker-01 192.168.83.101:80 check inter 1s

listen ingress-router-443
  bind *:443
  mode tcp
  balance source
    server worker-00 192.168.83.100:443 check inter 1s
    server worker-01 192.168.83.101:443 check inter 1s

listen ironic-api-6385
  bind *:6385
  mode tcp
  balance source
    server master-00 192.168.83.89:6385 check inter 1s
    server master-01 192.168.84.90:6385 check inter 1s
    server master-02 192.168.85.99:6385 check inter 1s
    server bootstrap 192.168.80.89:6385 check inter 1s

listen inspector-api-5050
  bind *:5050
  mode tcp
  balance source
    server master-00 192.168.83.89:5050 check inter 1s
    server master-01 192.168.84.90:5050 check inter 1s
    server master-02 192.168.85.99:5050 check inter 1s
    server bootstrap 192.168.80.89:5050 check inter 1s
# ...
```

2. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーとそのリソースが動作していることを確認します。

   a. 次のコマンドを実行して応答を観察し、クラスターマシン設定 API が Kubernetes API サーバーリソースにアクセスできることを確認します。

      ```
      $ curl https://<loadbalancer_ip_address>:6443/version --insecure
      ```

      設定が正しい場合は、応答として JSON オブジェクトを受信します。

      ```
      {
        "major": "1",
        "minor": "11+",
        "gitVersion": "v1.11.0+ad103ed",
        "gitCommit": "ad103ed",
        "gitTreeState": "clean",
        "buildDate": "2019-01-09T06:44:10Z",
        "goVersion": "go1.10.3",
      ```

```
  "compiler": "gc",
  "platform": "linux/amd64"
}
```

b. 次のコマンドを実行して出力を確認し、クラスターマシン設定 API がマシン設定サーバーリソースからアクセスできることを確認します。

```
$ curl -v https://<loadbalancer_ip_address>:22623/healthz --insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
Content-Length: 0
```

c. 次のコマンドを実行して出力を確認し、コントローラーがポート 80 の Ingress Controller リソースにアクセスできることを確認します。

```
$ curl -I -L -H "Host: console-openshift-console.apps.<cluster_name>.<base_domain>"
http://<load_balancer_front_end_IP_address>
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.ocp4.private.opequon.net/
cache-control: no-cache
```

d. 次のコマンドを実行して出力を確認し、コントローラーがポート 443 の Ingress Controller リソースにアクセスできることを確認します。

```
$ curl -I -L --insecure --resolve console-openshift-console.apps.<cluster_name>.
<base_domain>:443:<Load Balancer Front End IP Address> https://console-openshift-
console.apps.<cluster_name>.<base_domain>
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

2. ユーザー管理ロードバランサーのフロントエンド IP アドレスをターゲットにするようにクラス

3. ユーザー管理ロードバランサーのフロントエンド IP アドレスをターゲットにするようにクラスターの DNS レコードを設定します。ロードバランサー経由で、クラスター API およびアプリケーションの DNS サーバーのレコードを更新する必要があります。

変更された DNS レコードの例

```
<load_balancer_ip_address>  A  api.<cluster_name>.<base_domain>
A record pointing to Load Balancer Front End
```

```
<load_balancer_ip_address>   A apps.<cluster_name>.<base_domain>
A record pointing to Load Balancer Front End
```

重要

DNS の伝播では、各 DNS レコードが使用可能になるまでに時間がかかる場合があります。各レコードを検証する前に、各 DNS レコードが伝播されることを確認してください。

4. OpenShift Container Platform クラスターでユーザー管理ロードバランサーを使用するには、クラスターの **install-config.yaml** ファイルで次の設定を指定する必要があります。

```
# ...
platform:
  nutanix:
    loadBalancer:
      type: UserManaged (1)
      apiVIPs:
      - <api_ip> (2)
      ingressVIPs:
      - <ingress_ip> (3)
# ...
```

(1) クラスターのユーザー管理ロードバランサーを指定するには、**type** パラメーターに **UserManaged** を設定します。パラメーターのデフォルトは **OpenShiftManagedDefault** で、これはデフォルトの内部ロードバランサーを示します。**openshift-kni-infra** namespace で定義されたサービスの場合、ユーザー管理ロードバランサーは **coredns** サービスをクラスター内の Pod にデプロイできますが、**keepalived** および **haproxy** サービスは無視します。

(2) ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。Kubernetes API がユーザー管理ロードバランサーと通信できるように、ユーザー管理ロードバランサーのパブリック IP アドレスを指定します。

(3) ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。ユーザー管理ロードバランサーのパブリック IP アドレスを指定して、ユーザー管理ロードバランサーがクラスターの Ingress トラフィックを管理できるようにします。

検証

1. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーと DNS レコード設定が動作していることを確認します。

   a. 次のコマンドを実行して出力を確認し、クラスター API にアクセスできることを確認しま

す。

```
$ curl https://api.<cluster_name>.<base_domain>:6443/version --insecure
```

設定が正しい場合は、応答として JSON オブジェクトを受信します。

```
{
  "major": "1",
  "minor": "11+",
  "gitVersion": "v1.11.0+ad103ed",
  "gitCommit": "ad103ed",
  "gitTreeState": "clean",
  "buildDate": "2019-01-09T06:44:10Z",
  "goVersion": "go1.10.3",
  "compiler": "gc",
  "platform": "linux/amd64"
  }
```

b. 次のコマンドを実行して出力を確認し、クラスターマシン設定にアクセスできることを確認します。

```
$ curl -v https://api.<cluster_name>.<base_domain>:22623/healthz --insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
Content-Length: 0
```

c. 以下のコマンドを実行して出力を確認し、ポートで各クラスターアプリケーションにアクセスできることを確認します。

```
$ curl http://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.<cluster-name>.<base domain>/
cache-control: no-cacheHTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=39HoZgztDnzjJkq/JuLJMeoKNXlfiVv2YgZc09c3TBOBU4Nl6kDXaJH1LdicNhN1UsQ
Wzon4Dor9GWGfopaTEQ==; Path=/; Secure
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Tue, 17 Nov 2020 08:42:10 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=9b714eb87e93cf34853e87a92d6894be; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

d. 次のコマンドを実行して出力を確認し、ポート 443 で各クラスターアプリケーションにアクセスできることを確認します。

```
$ curl https://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

11.3.12. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

11.3.13. デフォルトのストレージコンテナーの設定

クラスターをインストールしたら、Nutanix CSI Operator をインストールし、クラスターのデフォルトのストレージコンテナーを設定する必要があります。

詳細は、CSI Operator のインストール と レジストリーストレージの設定 に関する Nutanix のドキュメントを参照してください。

11.3.14. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

11.3.15. 関連情報

- リモートヘルスモニタリングについて

11.3.16. 次のステップ

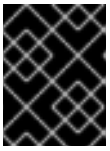
- リモートヘルスレポートのオプトアウト
- クラスターのカスタマイズ

11.4. ネットワークが制限された環境での NUTANIX へのクラスターのインストール

OpenShift Container Platform 4.16 では、インストールリリースコンテンツの内部ミラーを作成して、制限されたネットワーク内の Nutanix インフラストラクチャーにクラスターをインストールできます。

11.4.1. 前提条件

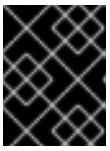
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- インストールプログラムで、Prism Central および Prism Element のポート 9440 にアクセスできる。ポート 9440 にアクセスできることを確認している。
- ファイアウォールを使用している場合は、次の前提条件を満たしています。
    - ポート 9440 にアクセスできることを確認している。コントロールプレーンノードがポート 9440 で Prism Central および Prism Element にアクセスできる (インストールが成功するために必要)。
    - OpenShift Container Platform が必要とするサイトへの アクセスを許可する ようにファイアウォールを設定している。これには、Telemetry の使用が含まれます。
- Nutanix 環境でデフォルトの自己署名 SSL/TLS 証明書を使用している場合は、CA によって署名された証明書に置き換える。インストールプログラムには、Prism Central API にアクセスするための有効な CA 署名付き証明書が必要です。自己署名証明書の置き換えに関する詳細は、Nutanix AOS Security Guide を参照してください。
  Nutanix 環境で内部 CA を使用して証明書を発行する場合は、インストールプロセスの一部としてクラスター全体のプロキシーを設定する必要があります。詳細は、カスタム PKI の設定 を参照してください。



重要

2048 ビット証明書を使用します。Prism Central 2022.x で 4096 ビット証明書を使用すると、インストールに失敗します。

- Red Hat Quay などのコンテナーイメージレジストリーがある。レジストリーがまだない場合は、**Red Hat OpenShift のミラーレジストリー** を使用してミラーレジストリーを作成できます。
- oc-mirror OpenShift CLI (oc) プラグイン を使用して、必要なすべての OpenShift Container Platform コンテンツと、Nutanix CSI Operator を含むその他のイメージをミラーレジストリーにミラーリングした。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

11.4.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

11.4.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

11.4.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

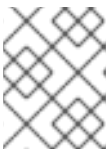
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

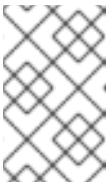
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

11.4.4. Nutanix root CA 証明書をシステム信頼に追加する

インストールプログラムは Prism Central API へのアクセスを必要とするため、OpenShift Container Platform クラスターをインストールする前に、Nutanix の信頼された root CA 証明書をシステム信頼に追加する必要があります。

手順

1. Prism Central Web コンソールから、Nutanix root CA 証明書をダウンロードします。
2. Nutanix root CA 証明書を含む圧縮ファイルを抽出します。
3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

11.4.5. RHCOS クラスターイメージのダウンロード

Prism Central は、クラスターをインストールするために Red Hat Enterprise Linux CoreOS (RHCOS) イメージにアクセスする必要があります。インストールプログラムを使用して、RHCOS イメージを見つけてダウンロードし、内部 HTTP サーバーまたは Nutanix オブジェクトを介して利用できるようにすることができます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install coreos print-stream-json
```

2. コマンドの出力を使用して Nutanix イメージの場所を見つけ、リンクをクリックしてダウンロードします。

出力例

```
"nutanix": {
  "release": "411.86.202210041459-0",
  "formats": {
    "qcow2": {
      "disk": {
        "location": "https://rhcos.mirror.openshift.com/art/storage/releases/rhcos-
4.11/411.86.202210041459-0/x86_64/rhcos-411.86.202210041459-0-
nutanix.x86_64.qcow2",
        "sha256":
"42e227cac6f11ac37ee8a2f9528bb3665146566890577fd55f9b950949e5a54b"
```

3. 内部 HTTP サーバーまたは Nutanix オブジェクトを介してイメージを利用できるようにします。

4. ダウンロードしたイメージの場所に注意してください。クラスターをデプロイする前に、インストール設定ファイル (**install-config.yaml**) の **platform** セクションをイメージの場所で更新します。

RHCOS イメージを指定する install-config.yaml ファイルのスニペット

```
platform:
  nutanix:
    clusterOSImage: http://example.com/images/rhcos-411.86.202210041459-0-
nutanix.x86_64.qcow2
```

11.4.6. インストール設定ファイルの作成

Nutanix にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- レジストリーをミラーリングしたときに作成された **imageContentSourcePolicy.yaml** ファイルを用意する。
- ダウンロードした Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所を用意する。
- ミラーレジストリーの証明書の内容を取得している。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、アクセス可能な場所にアップロードしました。
- Nutanix のネットワーク要件を満たしていることが確認されました。詳細については、Nutanix へのインストールの準備を参照してください。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットとするプラットフォームとして **nutanix** を選択します。

iii. Prism Central のドメイン名または IP アドレスを入力します。

iv. Prism Central へのログインに使用するポートを入力します。

v. Prism Central へのログインに使用する認証情報を入力します。
インストールプログラムが Prism Central に接続します。

vi. OpenShift Container Platform クラスターを管理する Prism Element を選択します。

vii. 使用するネットワークサブネットを選択します。

viii. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

ix. クラスター Ingress に設定した仮想 IP アドレスを入力します。

x. ベースドメインを入力します。このベースドメインは、DNS レコードで設定したものと同じである必要があります。

xi. クラスターの記述名を入力します。
入力するクラスター名は、DNS レコードの設定時に指定したクラスター名と一致する必要があります。

2. **install-config.yaml** ファイルで、**platform.nutanix.clusterOSImage** の値をイメージの場所または名前に設定します。以下に例を示します。

```
platform:
  nutanix:
    clusterOSImage: http://mirror.example.com/images/rhcos-47.83.202103221318-0-
nutanix.x86_64.qcow2
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
```

```
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
-----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、レジストリーをミラーリングしたときに作成された **imageContentSourcePolicy.yaml** ファイルを使用します。

d. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

4. オプション: **install.config.yaml** ファイル内の 1 つ以上のデフォルト設定パラメーターを更新して、インストールをカスタマイズします。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

注記

3 ノードクラスターをインストールする場合は、必ず **compute.replicas** パラメーターを **0** に設定してください。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細は、「{platform} への 3 ノードクラスターのインストール」を参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- Nutanix のインストール設定パラメーター

11.4.6.1. Nutanix 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    nutanix: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMiB: 8196
      osDisk:
        diskSizeGiB: 120
      categories: (5)
      - key: <category_key_name>
        value: <category_value>
controlPlane: (6)
  hyperthreading: Enabled (7)
  name: master
  replicas: 3
  platform:
    nutanix: (8)
      cpus: 4
      coresPerSocket: 2
      memoryMiB: 16384
      osDisk:
        diskSizeGiB: 120
      categories: (9)
      - key: <category_key_name>
        value: <category_value>
metadata:
  creationTimestamp: null
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (11)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  nutanix:
    apiVIP: 10.40.142.7 (12)
```

```
    ingressVIP: 10.40.142.8 13
    defaultMachinePlatform:
      bootType: Legacy
      categories: 14
      - key: <category_key_name>
        value: <category_value>
      project: 15
        type: name
        name: <project_name>
    prismCentral:
      endpoint:
        address: your.prismcentral.domainname 16
        port: 9440 17
      password: <password> 18
      username: <username> 19
    prismElements:
    - endpoint:
        address: your.prismelement.domainname
        port: 9440
      uuid: 0005b0f1-8f43-a0f2-02b7-3cecef193712
    subnetUUIDs:
    - c7938dc6-7659-453e-a688-e26020c68e43
    clusterOSImage: http://example.com/images/rhcos-47.83.202103221318-0-nutanix.x86_64.qcow2
20
credentialsMode: Manual
publish: External
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 21
fips: false 22
sshKey: ssh-ed25519 AAAA... 23
additionalTrustBundle: | 24
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 25
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 10 12 13 16 17 18 19 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチ

スレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 8 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

5 9 14 オプション: プリズムカテゴリーキーとプリズムカテゴリー値のペアを 1 つ以上指定します。これらのカテゴリーのキーと値のペアは、Prism Central に存在する必要があります。コンピューティングマシン、コントロールプレーンマシン、またはすべてのマシンに個別のカテゴリーを指定できます。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

15 オプション: VM が関連付けられているプロジェクトを指定します。プロジェクトタイプの **name** または **uuid** を指定してから、対応する UUID またはプロジェクト名を指定します。プロジェクトは、コンピューティングマシン、コントロールプレーンマシン、またはすべてのマシンに関連付けることができます。

20 オプション: デフォルトでは、インストールプログラムは Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードしてインストールします。Prism Central がインターネットにアクセスできない場合は、任意の HTTP サーバーまたは Nutanix オブジェクトで RHCOS イメージをホストし、インストールプログラムがそのイメージを指すようにすることで、デフォルトの動作をオーバーライドできます。

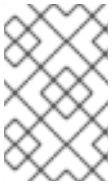
21 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

22 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

23 オプション: クラスター内のマシンにアクセスするのに使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

24 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

25 レジストリーをミラーリングしたときに作成された **imageContentSourcePolicy.yaml** ファイルの **metadata.name: release-0** セクションからこれらの値を指定します。

11.4.6.2. 障害ドメインの設定

障害ドメインは、コントロールプレーンとコンピュートマシンを複数の Nutanix Prism Element (クラスター) に分散することで、OpenShift Container Platform クラスターのフォールトトレランスを向上させます。

ヒント

高可用性を確保するには、3 つの障害ドメインを設定することを推奨します。

前提条件

- インストール設定ファイル (**install-config.yaml**) がある。

手順

1. **install-config.yaml** ファイルを編集し、次のスタンザを追加して最初の障害ドメインを設定します。

```
apiVersion: v1
baseDomain: example.com
compute:
# ...
platform:
  nutanix:
    failureDomains:
    - name: <failure_domain_name>
      prismElement:
        name: <prism_element_name>
        uuid: <prism_element_uuid>
      subnetUUIDs:
      - <network_uuid>
# ...
```

ここでは、以下のようになります。

<failure_domain_name>

障害ドメインの一意の名前を指定します。名前は 64 文字以下に制限されており、小文字、数字、ダッシュ (-) を含めることができます。ダッシュを名前の先頭または末尾に含めることはできません。

<prism_element_name>

オプション: Prism Element の名前を指定します。

<prism_element_uuid>

Prism Element の UUID を指定します。

<network_uuid>

Prism Element サブネットオブジェクトの UUID を指定します。サブネットの IP アドレス接

頭辞 (CIDR) には、OpenShift Container Platform クラスターが使用する仮想 IP アドレスを含める必要があります。OpenShift Container Platform クラスター内の障害ドメイン (Prism Element) ごとに 1 つのサブネットのみがサポートされます。

2. 必要に応じて、追加の障害ドメインを設定します。

3. コントロールプレーンとコンピュートマシンを障害ドメイン全体に分散するには、次のいずれかを実行します。

   - コンピューティングプレーンとコントロールプレーンのマシンが同じ障害ドメインセットを共有できる場合は、クラスターのデフォルトのマシン設定の下に障害ドメイン名を追加します。

     障害ドメインセットを共有するコントロールプレーンとコンピュートマシンの例

     ```
     apiVersion: v1
     baseDomain: example.com
     compute:
     # ...
     platform:
       nutanix:
         defaultMachinePlatform:
           failureDomains:
             - failure-domain-1
             - failure-domain-2
             - failure-domain-3
     # ...
     ```

   - コンピュートマシンとコントロールプレーンマシンが別々の障害ドメインを使用する必要がある場合は、それぞれのマシンプールの下に障害ドメイン名を追加します。

     別々の障害ドメインを使用するコントロールプレーンとコンピュートマシンの例

     ```
     apiVersion: v1
     baseDomain: example.com
     compute:
     # ...
     controlPlane:
       platform:
         nutanix:
           failureDomains:
             - failure-domain-1
             - failure-domain-2
             - failure-domain-3
     # ...
     compute:
       platform:
         nutanix:
           failureDomains:
             - failure-domain-1
             - failure-domain-2
     # ...
     ```

4. ファイルを保存します。

11.4.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを

openshift-config namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

11.4.7. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

11.4.8. Nutanix の IAM の設定

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムが手動モード用に CCO を設定する間、ID およびアクセス管理シークレットを指定する必要があります。

前提条件

- **ccoctl** バイナリーを設定している。
- **install-config.yaml** ファイルがある。

手順

1. 認証情報データを含む YAML ファイルを次の形式で作成します。

認証情報データの形式

```
credentials:
- type: basic_auth (1)
  data:
    prismCentral: (2)
      username: <username_for_prism_central>
```

```
    password: <password_for_prism_central>
  prismElements: 3
  - name: <name_of_prism_element>
    username: <username_for_prism_element>
    password: <password_for_prism_element>
```

1 認証タイプを指定します。Basic 認証のみがサポートされています。

2 Prism Central の認証情報を指定します。

3 オプション: Prism Element 認証情報を指定します。

2. 次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

2 **install-config.yaml** ファイルの場所を指定します。

3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  annotations:
    include.release.openshift.io/self-managed-high-availability: "true"
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-machine-api-nutanix
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: NutanixProviderSpec
  secretRef:
    name: nutanix-credentials
    namespace: openshift-machine-api
```

-

4. 次のコマンドを実行し、**ccoctl** ツールを使用して **CredentialsRequest** オブジェクトをすべて処理します。

```
$ ccoctl nutanix create-shared-secrets \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 1
  --output-dir=<ccoctl_output_dir> \ 2
  --credentials-source-filepath=<path_to_credentials_file> 3
```

1. コンポーネント **CredentialsRequests** オブジェクトのファイルを含むディレクトリーへのパスを指定します。
2. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
3. オプション: 認証情報データ YAML ファイルを含むディレクトリーを指定します。デフォルトでは、**ccoctl** はこのファイルが **<home_directory>/.nutanix/credentials** にあると想定します。

5. **credentialsMode** パラメーターが **Manual** に設定されるように、**install-config.yaml** 設定ファイルを編集します。

サンプル install-config.yaml 設定ファイル

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual 1
...
```

1. この行を追加して、**credentialsMode** パラメーターを **Manual** に設定します。

6. 次のコマンドを実行して、インストールマニフェストを作成します。

```
$ openshift-install create manifests --dir <installation_directory> 1
```

1. クラスターの **install-config.yaml** ファイルを含むディレクトリーへのパスを指定します。

7. 次のコマンドを実行して、生成された認証情報ファイルをターゲットマニフェストディレクトリーにコピーします。

```
$ cp <ccoctl_output_dir>/manifests/*credentials.yaml ./<installation_directory>/manifests
```

検証

- **manifests** ディレクトリーに適切なシークレットが存在することを確認します。

```
$ ls ./<installation_directory>/manifests
```

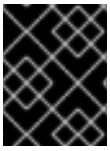
出力例

-

```
cluster-config.yaml
cluster-dns-02-config.yml
cluster-infrastructure-02-config.yml
cluster-ingress-02-config.yml
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-proxy-01-config.yaml
cluster-scheduler-02-config.yml
cvo-overrides.yaml
kube-cloud-config.yaml
kube-system-configmap-root-ca.yaml
machine-config-server-tls-secret.yaml
openshift-config-secret-pull-secret.yaml
openshift-cloud-controller-manager-nutanix-credentials-credentials.yaml
openshift-machine-api-nutanix-credentials-credentials.yaml
```

11.4.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。

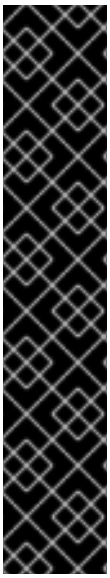
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

11.4.10. インストール後の設定

以下の手順を実行して、クラスターの設定を完了します。

11.4.10.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

11.4.10.2. クラスターへのポリシーリソースのインストール

oc-mirror OpenShift CLI (oc) プラグインを使用して OpenShift Container Platform コンテンツをミラーリングすると、**catalogSource-certified-operator-index.yaml** および **imageContentSourcePolicy.yaml** を含むリソースが作成されます。

- **ImageContentSourcePolicy** リソースは、ミラーレジストリーをソースレジストリーに関連付け、イメージプル要求をオンラインレジストリーからミラーレジストリーにリダイレクトします。
- **CatalogSource** リソースは、Operator Lifecycle Manager (OLM) によって使用され、ミラーレジストリーで使用可能な Operator に関する情報を取得します。これにより、ユーザーは Operator を検出してインストールできます。

クラスターをインストールしたら、これらのリソースをクラスターにインストールする必要があります。

前提条件

- 非接続環境で、イメージセットをレジストリーミラーにミラーリングしました。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift CLI にログインします。
2. results ディレクトリーからクラスターに YAML ファイルを適用します。

```
$ oc apply -f ./oc-mirror-workspace/results-<id>/
```

検証

1. **ImageContentSourcePolicy** リソースが正常にインストールされたことを確認します。

```
$ oc get imagecontentsourcepolicy
```

2. **CatalogSource** リソースが正常にインストールされたことを確認します。

```
$ oc get catalogsource --all-namespaces
```

11.4.10.3. デフォルトのストレージコンテナーの設定

クラスターをインストールしたら、Nutanix CSI Operator をインストールし、クラスターのデフォルトのストレージコンテナーを設定する必要があります。

詳細は、CSI Operator のインストール と レジストリーストレージの設定 に関する Nutanix のドキュメントを参照してください。

11.4.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

11.4.12. 関連情報

- リモートヘルスモニタリングについて

11.4.13. 次のステップ

- 必要に応じて、リモートヘルスレポートのオプトアウト を参照してください。
- 必要に応じて、非接続クラスターの登録 を参照してください。
- クラスターのカスタマイズ

11.5. NUTANIX への 3 ノードクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、Nutanix に 3 ノードクラスターをインストールできます。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコントロールプレーンマシンで設定されます。このタイプのクラスターは、クラスター管理者および開発者がテスト、開発、および実稼働に使用するためのより小さくリソース効率の高いクラスターを提供します。

11.5.1. 3 ノードクラスターの設定

クラスターをデプロイする前に、**install-config.yaml** ファイルでワーカーノードの数を **0** に設定して、3 ノードクラスターを設定します。ワーカーノードの数を **0** に設定すると、コントロールプレーンマシンがスケジュール可能になります。これにより、アプリケーションワークロードをコントロールプレーンノードから実行するようにスケジュールできます。

注記

アプリケーションワークロードはコントロールプレーンノードから実行され、コントロールプレーンノードはコンピュートノードと見なされるため、追加のサブスクリプションが必要です。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 次の **compute** スタンザに示すように、**install-config.yaml** ファイルでコンピューティングレプリカの数を **0** に設定します。

  3 ノードクラスターの install-config.yaml ファイルの例

```
apiVersion: v1
baseDomain: example.com
compute:
- name: worker
  platform: {}
  replicas: 0
# ...
```

11.5.2. 次のステップ

- クラスターの Nutanix へのインストール

11.6. NUTANIX でのクラスターのアンインストール

Nutanix にデプロイしたクラスターを削除できます。

11.6.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

11.7. NUTANIX のインストール設定パラメーター

OpenShift Container Platform クラスターを Nutanix にデプロイする前に、クラスターとそれをホストするプラットフォームをカスタマイズするためのパラメーターを指定します。**install-config.yaml** ファイルを作成するときは、コマンドラインを使用して必要なパラメーターの値を指定します。その後、**install-config.yaml** ファイルを変更して、クラスターをさらにカスタマイズできます。

11.7.1. Nutanix で使用可能なインストール設定パラメーター

次の表では、インストールプロセスの一部として設定できる、必須、オプション、および Nutanix 固有のインストール設定パラメーターを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

11.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表11.3 必須パラメーター

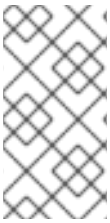
パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String

パラメーター	説明	値
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

11.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。



注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表11.4 ネットワークパラメーター

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。

パラメーター	説明	値
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

11.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表11.5 オプションのパラメーター

パラメーター	説明	値

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**

パラメーター	説明	値
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]

パラメーター	説明	値
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。 **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。 **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。 **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

11.7.1.4. 追加の Nutanix 設定パラメーター

追加の Nutanix 設定パラメーターについては、次の表で説明します。

表11.6 追加の Nutanix クラスターパラメーター

パラメーター	説明	値
compute: platform: nutanix: categories: key:	コンピューティング VM に適用するプリズムカテゴリーのキーの名前。このパラメーターには、**value** パラメーターを指定する必要があり、**key** と **value** の両方のパラメーターが Prism Central に存在する必要があります。カテゴリーの詳細については、カテゴリー管理 を参照してください。	String
compute: platform: nutanix: categories: value:	コンピューティング VM に適用するプリズムカテゴリーのキーと値のペアの値。このパラメーターには、**key** パラメーターを指定する必要があり、**key** と **value** の両方のパラメーターが Prism Central に存在する必要があります。	String
compute: platform: nutanix: failureDomains:	コンピュートマシンのみに適用する障害ドメイン。 障害ドメインは **platform.nutanix.failureDomains** で指定します。	List。 1 つ以上の障害ドメインの名前。
compute: platform: nutanix: project: type:	コンピューティング VM のプロジェクトを選択するために使用する識別子のタイプ。プロジェクトは、権限、ネットワーク、およびその他のパラメーターを管理するためのユーザーロールの論理グループを定義します。プロジェクトの詳細については、プロジェクトの概要 を参照してください。	**name** または **uuid**
compute: platform: nutanix: project: name: or uuid:	コンピューティング VM が関連付けられているプロジェクトの名前または UUID。このパラメーターには、**type** パラメーターを指定する必要があります。	String

パラメーター	説明	値
compute: platform: nutanix: bootType:	コンピューティングマシンが使用するブートタイプ。OpenShift Container Platform 4.16 では、**Legacy** ブートタイプを使用する必要があります。ブートタイプの詳細については、仮想化環境内の UEFI、セキュアブート、および TPM について を参照してください。	**Legacy**、**SecureBoot**、または **UEFI**。デフォルトは、**Legacy** です。
controlPlane: platform: nutanix: categories: key:	コントロールプレーン VM に適用するプリズムカテゴリーのキーの名前。このパラメーターには、**value** パラメーターを指定する必要があり、**key** と **value** の両方のパラメーターが Prism Central に存在する必要があります。カテゴリーの詳細については、カテゴリー管理 を参照してください。	String
controlPlane: platform: nutanix: categories: value:	コントロールプレーン VM に適用するプリズムカテゴリーのキーと値のペアの値。このパラメーターには、**key** パラメーターを指定する必要があり、**key** と **value** の両方のパラメーターが Prism Central に存在する必要があります。	String
controlPlane: platform: nutanix: failureDomains:	コントロールプレーンマシンのみに適用する障害ドメイン。 障害ドメインは **platform.nutanix.failureDomains** で指定します。	List。 1 つ以上の障害ドメインの名前。
controlPlane: platform: nutanix: project: type:	コントロールプレーン VM のプロジェクトを選択するために使用する識別子のタイプ。プロジェクトは、権限、ネットワーク、およびその他のパラメーターを管理するためのユーザーロールの論理グループを定義します。プロジェクトの詳細については、プロジェクトの概要 を参照してください。	**name** または **uuid**
controlPlane: platform: nutanix: project: name: or uuid:	コントロールプレーン VM が関連付けられているプロジェクトの名前または UUID。このパラメーターには、**type** パラメーターを指定する必要があります。	String

パラメーター	説明	値
platform: nutanix: defaultMachinePlatform: categories: key:	すべての VM に適用するプリズムカテゴリーのキーの名前。このパラメーターには、**value** パラメーターを指定する必要があり、**key** と **value** の両方のパラメーターが Prism Central に存在する必要があります。カテゴリーの詳細については、カテゴリー管理 を参照してください。	String
platform: nutanix: defaultMachinePlatform: categories: value:	すべての VM に適用するプリズムカテゴリーのキーと値のペアの値。このパラメーターには、**key** パラメーターを指定する必要があり、**key** と **value** の両方のパラメーターが Prism Central に存在する必要があります。	String
platform: nutanix: defaulatMachinePlatform: failureDomains:	コントロールプレーンとコンピュートマシンの両方に適用する障害ドメイン。 障害ドメインは **platform.nutanix.failureDomains** で指定します。	List。 1 つ以上の障害ドメインの名前。
platform: nutanix: defaultMachinePlatform: project: type:	すべての VM のプロジェクトを選択するために使用する識別子のタイプ。プロジェクトは、権限、ネットワーク、およびその他のパラメーターを管理するためのユーザーロールの論理グループを定義します。プロジェクトの詳細については、プロジェクトの概要 を参照してください。	**name** または **uuid**。
platform: nutanix: defaultMachinePlatform: project: name: or uuid:	すべての VM が関連付けられているプロジェクトの名前または UUID。このパラメーターには、**type** パラメーターを指定する必要があります。	String

パラメーター	説明	値
platform: nutanix: defaultMachinePlatform: bootType:	すべてのマシンのブートタイプ。OpenShift Container Platform 4.16 では、**Legacy** ブートタイプを使用する必要があります。ブートタイプの詳細については、仮想化環境内の UEFI、セキュアブート、および TPM についてを参照してください。	**Legacy**、**SecureBoot**、または **UEFI**。デフォルトは、**Legacy** です。
platform: nutanix: apiVIP:	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス
platform: nutanix: failureDomains: - name: prismElement: name: uuid: subnetUUIDs: -	デフォルトでは、インストールプログラムはクラスターマシンを単一の Prism Element インスタンスにインストールします。フォールトトレランスのために追加の Prism Element インスタンスを指定し、そのインスタンスを次のものに適用できます。 • クラスターのデフォルトのマシン設定 • コントロールプレーンまたはコンピュートマシンプールのみ	設定した障害ドメインのリスト。 使用方法の詳細は、「Nutanix へのクラスターのインストール」の「障害ドメインの設定」を参照してください。
platform: nutanix: ingressVIP:	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス
platform: nutanix: prismCentral: endpoint: address:	Prism Central ドメイン名または IP アドレス。	String
platform: nutanix: prismCentral: endpoint: port:	Prism Central へのログインに使用されるポート。	String

パラメーター	説明	値
platform: nutanix: prismCentral: password:	Prism Central ユーザー名のパスワード。	String
platform: nutanix: prismCentral: username:	Prism Central へのログインに使用されるユーザー名。	String
platform: nutanix: prismElements: endpoint: address:	Prism Element ドメイン名または IP アドレス。 [1]	String
platform: nutanix: prismElements: endpoint: port:	Prism Element へのログインに使用されるポート。	String
platform: nutanix: prismElements: uuid:	Prism Element の Universally Unique Identifier (UUID)。	String
platform: nutanix: subnetUUIDs:	設定した仮想 IP アドレスと DNS レコードを含む Prism Element ネットワークの UUID。 [2]	String
platform: nutanix: clusterOSImage:	オプション: デフォルトでは、インストールプログラムは Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードしてインストールします。Prism Central がインターネットにアクセスできない場合は、任意の HTTP サーバーで RHCOS イメージをホストし、インストールプログラムがそのイメージを指すようにすることで、デフォルトの動作をオーバーライドできます。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: http://example.com/images/rhcos-47.83.202103221318-0-nutanix.x86_64.qcow2

1. **prismElements** セクションには、Prism Elements (クラスター) のリストが含まれています。Prism Element は、OpenShift Container Platform クラスターをホストするために使用されるすべての Nutanix リソース (仮想マシンやサブネットなど) を包含します。

2. OpenShift Container Platform クラスター内の Prism Element ごとに 1 つのサブネットのみがサポートされます。

第12章 アシステッドインストーラーを使用したオンプレミスのインストール

12.1. ASSISTED INSTALLER を使用したオンプレミスクラスターのインストール

Assisted Installer を使用して、OpenShift Container Platform をオンプレミスのハードウェアまたはオンプレミスの VM にインストールできます。Assisted Installer を使用して OpenShift Container Platform をインストールすると、**x86_64**、**AArch64**、**ppc64le**、および **s390x** アーキテクチャーがサポートされます。

12.1.1. Assisted Installer の使用

Assisted Installer は、Red Hat Hybrid Cloud Console で提供されるユーザーフレンドリーなインストールソリューションです。Assisted Installer は、ベアメタル、Nutanix、および vSphere インフラストラクチャーを重視するさまざまなデプロイメントプラットフォームをサポートします。

Assisted Installer は、インストール機能をサービスとして提供します。このサービスとしてのソフトウェア (SaaS) アプローチには、次の利点があります。

- **Web ユーザーインターフェイス:** Web ユーザーインターフェイスは、ユーザーがインストール設定ファイルを手動で作成しなくても、クラスターのインストールを実行します。
- **ブートストラップノードなし:** Assisted Installer を使用してインストールする場合、ブートストラップノードは必要ありません。ブートストラッププロセスは、クラスター内のノードで実行されます。
- **ホスティング:** Assisted Installer は以下をホストします。
    - Ignition ファイル
    - インストール前の設定
    - 検出 ISO
    - インストーラー
- **合理化されたインストールワークフロー:** デプロイメントには、OpenShift Container Platform の詳細な知識は必要ありません。Assisted Installer は適切なデフォルトを提供し、以下のようにインストーラーをサービスとして提供します。
    - OpenShift Container Platform インストーラーをローカルにインストールして実行する必要がなくなります。
    - 最新のテスト済み z-stream リリースまでの最新バージョンのインストーラーを保証します。必要に応じて、古いバージョンを引き続き利用できます。
    - OpenShift Container Platform インストーラーをローカルで実行する必要なく、API を使用したビルドの自動化を可能にします。
- **高度なネットワーキング:** Assisted Installer は、SDN と OVN を使用する IPv4 ネットワーキング、OVN のみを使用する IPv6 とデュアルスタックネットワーキング、NMState ベースの静的 IP アドレス指定、および HTTP/S プロキシーをサポートします。OVN は、OpenShift

Container Platform 4.12 以降のリリースにおけるデフォルトの Container Network Interface (CNI) です。SDN は OpenShift Container Platform 4.14 以前でサポートされますが、OpenShift Container Platform 4.15 以降のリリースではサポートされていません。

- **インストール前の検証:** Assisted Installer は、インストール前に設定を検証して、高い確率で成功するようにします。検証プロセスには次のチェックが含まれます。
  - ネットワーク接続の確保
  - 十分なネットワーク帯域幅の確保
  - レジストリーへの接続の確保
  - クラスターノード間の時刻同期の確保
  - クラスターノードが最小ハードウェア要件を満たしていることの確認
  - インストール設定パラメーターの検証
- **REST API:** Assisted Installer には REST API があり、自動化が可能となります。

Assisted Installer は、オプションの HTTP/S プロキシーを含む、接続された環境でのオンプレミスの OpenShift Container Platform のインストールをサポートします。以下をインストールできます。

- 高可用性 OpenShift Container Platform またはシングルノード OpenShift (SNO)
- プラットフォームが完全に統合されたベアメタル、Nutanix、または vSphere 上の OpenShift Container Platform、または統合されていないその他の仮想化プラットフォーム
- オプション: OpenShift Virtualization、マルチクラスターエンジン、論理ボリュームマネージャー (LVM) ストレージ、OpenShift Data Foundation

注記

現在、OpenShift Virtualization および LVM ストレージは IBM Z® (**s390x**) アーキテクチャーでサポートされていません。

ユーザーインターフェイスは、自動化が存在しない、または必要とされない直感的なインタラクティブワークフローを提供します。ユーザーは、REST API を使用してインストールを自動化することもできます。

詳細は、OpenShift Container Platform の Assisted Installer のドキュメントを参照してください。

12.1.2. Assisted Installer の API サポート

アシステッドインストーラーの API サポートは、非推奨の発表から少なくとも 3 カ月は安定しています。

第13章 AGENT-BASED INSTALLER を使用したオンプレミスクラスターのインストール

13.1. AGENT-BASED INSTALLER を使用したインストールの準備

13.1.1. Agent-based Installer について

エージェントベースのインストール方法では、選択した任意の方法でオンプレミスサーバーを柔軟に起動できます。Assisted Installation サービスの使いやすさと、エアギャップ環境を含むオフラインでの実行機能を兼ね備えています。エージェントベースのインストールは、OpenShift Container Platform インストーラーのサブコマンドです。OpenShift Container Platform クラスターをデプロイするために必要なすべての情報を含む起動可能な ISO イメージを、利用可能なリリースイメージと共に生成します。

設定は、インストーラーによってプロビジョニングされたインフラストラクチャーおよびユーザーによってプロビジョニングされたインフラストラクチャーのインストール方法と同じ形式です。Agent-based Installer は、必要に応じてゼロタッチプロビジョニング (ZTP) カスタムリソースを生成または受け入れることもできます。ZTP を使用すると、ベアメタル機器の宣言型設定で新しいエッジサイトをプロビジョニングできます。

表13.1 Agent-based Installer でサポートされるアーキテクチャー

CPU アーキテクチャー	接続インストール	非接続インストール
64-bit x86	✓	✓
64-bit ARM	✓	✓
ppc64le	✓	✓
s390x	✓	✓

13.1.2. Agent-based Installer について

OpenShift Container Platform ユーザーは、切断された環境でアシステッドインストーラーのホストサービスの利点を活用できます。

エージェントベースのインストールは、Assisted Discovery Agent と Assisted Service を含む起動可能な ISO で構成されます。クラスターのインストールを実行するには両方が必要ですが、後者はいずれかのホストでのみ実行されます。

openshift-install agent create image サブコマンドは、指定した入力をもとに一時 ISO を生成します。以下のマニフェストで入力を指定できます。

推奨:

- **install-config.yaml**
- **agent-config.yaml**

オプション: ZTP マニフェスト

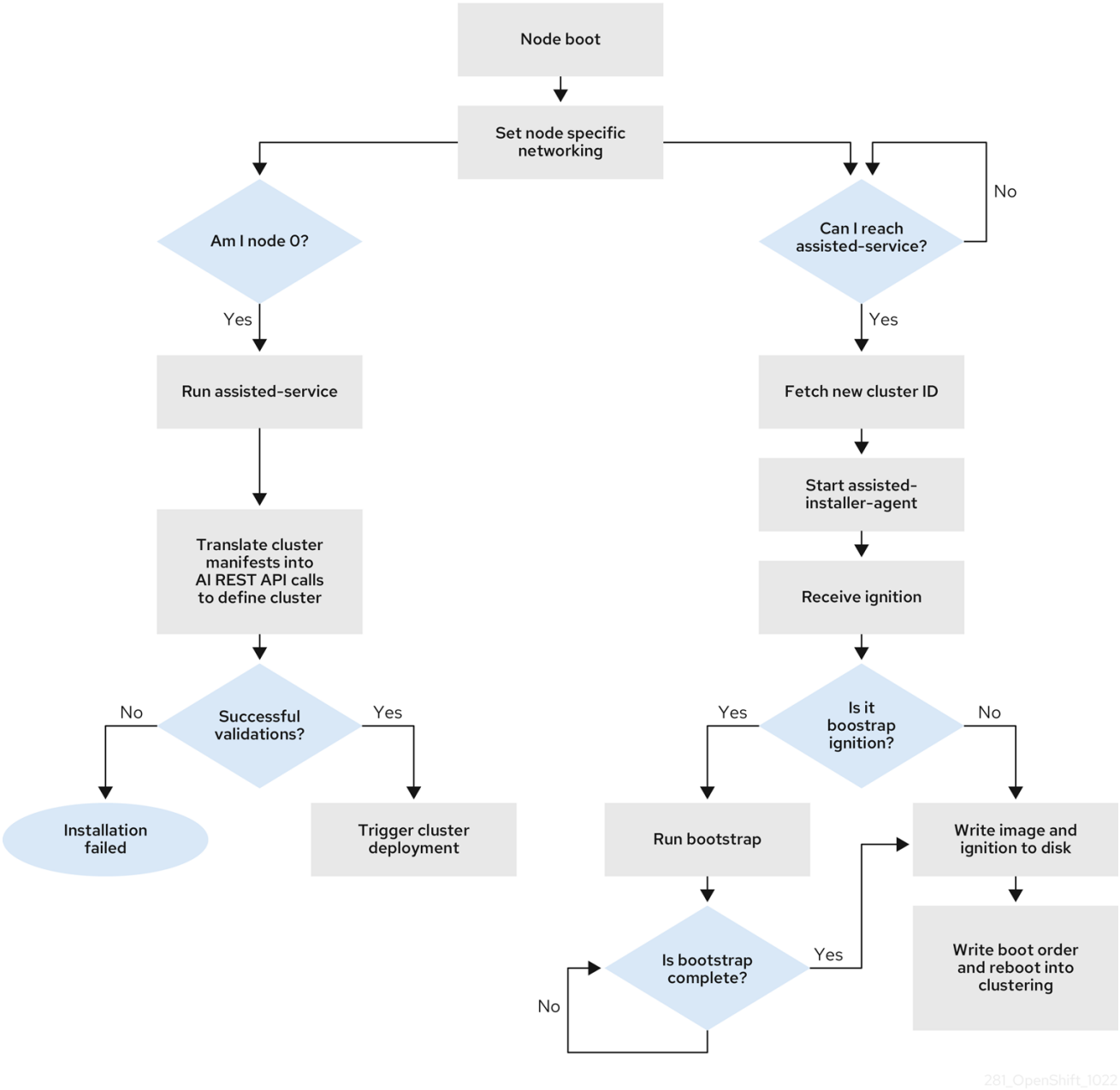
- **cluster-manifests/cluster-deployment.yaml**

- **cluster-manifests/agent-cluster-install.yaml**
- **cluster-manifests/pull-secret.yaml**
- **cluster-manifests/infraenv.yaml**
- **cluster-manifests/cluster-image-set.yaml**
- **cluster-manifests/nmstateconfig.yaml**
- **mirror/registries.conf**
- **mirror/ca-bundle.crt**

13.1.2.1. Agent-based Installer のワークフロー

コントロールプレーンホストの 1 つは、ブートプロセスの開始時に Assisted Service を実行し、最終的にブートストラップホストになります。このノードを **ランデブーホスト** (ノード 0) と呼びます。Assisted Service は、すべてのホストが要件を満たしていることを確認し、OpenShift Container Platform クラスターのデプロイをトリガーします。すべてのノードで Red Hat Enterprise Linux CoreOS (RHCOS) イメージがディスクに書き込まれます。ブートストラップ以外のノードは再起動し、クラスターデプロイメントを開始します。ノードが再起動されると、ランデブーホストが再起動し、クラスターに参加します。ブートストラップが完了し、クラスターがデプロイされます。

図13.1 ノードのインストールワークフロー



以下のトポロジーでは、**openshift-install agent create image** サブコマンドを使用して、ネットワークに接続されていない OpenShift Container Platform クラスターをインストールできます。

- **単一ノードの OpenShift Container Platform クラスター(SNO)**: マスターとワーカーの両方であるノード。
- **3 ノードの OpenShift Container Platform クラスター**: ワーカーノードでもある 3 つのマスターノードを持つコンパクトなクラスター。
- **高可用性 OpenShift Container Platform クラスター(HA)**: 任意の数のワーカーノードを持つ 3 つのマスターノード。

13.1.2.2. 各トポロジーに推奨されるリソース

以下の各トポロジーに推奨されるクラスターリソースを示します。

表13.2 推奨されるクラスターリソース

トポロジー	コントロールプレーンノードの数	コンピュートノードの数	vCPU	メモリー	ストレージ
シングルノードクラスター	1	0	8 vCPU コア	16 GB のメモリー	120 GB
コンパクトなクラスター	3	0 または 1	8 vCPU コア	16 GB のメモリー	120 GB
HA クラスター	3	2 以降	8 vCPU コア	16 GB のメモリー	120 GB

install-config.yaml で、インストールを実行するプラットフォームを指定します。以下のプラットフォームがサポートされます。

- **baremetal**
- **vsphere**
- **none**

重要

プラットフォームが **none** の場合:

- **none** オプションでは、クラスター内に DNS 名前解決と負荷分散インフラストラクチャーをプロビジョニングする必要があります。詳細は、「関連情報」セクションの **プラットフォーム "none" オプションを使用するクラスターの要件** 参照してください。
- 仮想化またはクラウド環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

関連情報

- プラットフォーム "none" オプションを使用するクラスターの要件
- ネットワーク MTU の増加
- シングルノード OpenShift クラスターへのワーカーノードの追加

13.1.3. FIPS 準拠について

多くの OpenShift Container Platform のお客様においては、システムが実稼働環境で使用される前に、一定レベルでの規制への対応またはコンプライアンスが必要になります。この規制対応は、国家標準、業界標準または組織の企業ガバナンスフレームワークによって課せられます。FIPS (Federal Information Processing Standards) コンプライアンスは、安全な環境で必要とされる最も重要なコンポーネントの 1 つであり、サポートされている暗号化技術のみがノード上で許可されるようにします。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

13.1.4. Agent-based Installer による FIPS の設定

クラスターのデプロイ中に、Red Hat Enterprise Linux CoreOS (RHCOS) マシンがクラスターにデプロイされると、連邦情報処理標準 (FIPS) の変更が適用されます。Red Hat Enterprise Linux (RHEL) マシンでは、ワーカーマシンとして使用する予定のマシンにオペレーティングシステムをインストールする場合に FIPS モードを有効にする必要があります。

install-config.yaml および **agent-config.yaml** の推奨される方法で FIPS モードを有効にできます。

1. **install-config.yaml** ファイルで **fips** フィールドの値を **True** に設定する必要があります。

   サンプル install-config.yaml.file

   ```
   apiVersion: v1
   baseDomain: test.example.com
   metadata:
     name: sno-cluster
   fips: True
   ```

2. オプション: GitOps ZTP マニフェストを使用している場合、**agent-cluster-install.yaml** ファイルの **Agent-install.openshift.io/install-config-overrides** フィールドで **fips** の値を **True** に設定する必要があります。

   サンプルの agent-cluster-install.yaml ファイル

   ```
   apiVersion: extensions.hive.openshift.io/v1beta1
   kind: AgentClusterInstall
   metadata:
     annotations:
       agent-install.openshift.io/install-config-overrides: '{"fips": True}'
     name: sno-cluster
     namespace: sno-cluster-test
   ```

関連情報

- OpenShift セキュリティーガイドブック
- FIPS 暗号のサポート

13.1.5. ホストの設定

クラスター上の各ホストの **agent-config.yaml** ファイルで、ネットワーク設定やルートデバイスのヒントなど追加設定を行うことができます。

重要

設定するホストごとに、ホスト上のインターフェイスの MAC アドレスで、設定するホストを指定する必要があります。

13.1.5.1. ホストのロール

クラスター内の各ホストには、**master** または **worker** のいずれかのロールが割り当てられます。**role** パラメーターを使用して、**agent-config.yaml** ファイルで各ホストのロールを定義できます。ホストにロールを割り当てない場合、インストール時にロールがランダムに割り当てられます。

そのため、ホストのロールは明示的に定義することが推奨されます。

rendezvousIP は、**master** ロールを持つホストに割り当てる必要があります。これは、手動で、または Agent-based Installer によるロールの割り当てを許可することで行えます。

重要

ランデブーホストの **master** ロールを明示的に定義する必要はありませんが、この割り当てと競合する設定は作成できません。

たとえば、ホストが 4 台あり、そのうち 3 台が **master** ロールを持つように明示的に定義されている場合、最後の 1 台にはインストール時に **worker** ロールが自動的に割り当てられ、ランデブーホストには設定できません。

agent-config.yaml のサンプルファイル

```
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: example-cluster
rendezvousIP: 192.168.111.80
hosts:
  - hostname: master-1
    role: master
    interfaces:
      - name: eno1
        macAddress: 00:ef:44:21:e6:a5
  - hostname: master-2
    role: master
    interfaces:
      - name: eno1
        macAddress: 00:ef:44:21:e6:a6
  - hostname: master-3
    role: master
    interfaces:
      - name: eno1
        macAddress: 00:ef:44:21:e6:a7
  - hostname: worker-1
    role: worker
```

```
interfaces:
  - name: eno1
    macAddress: 00:ef:44:21:e6:a8
```

13.1.5.2. ルートデバイスヒントについて

rootDeviceHints パラメーターは、インストーラーが Red Hat Enterprise Linux CoreOS (RHCOS) イメージを特定のデバイスにプロビジョニングできるようにします。インストーラーは、検出順にデバイスを検査し、検出された値をヒントの値と比較します。インストーラーは、ヒント値に一致する最初に検出されたデバイスを使用します。この設定は複数のヒントを組み合わせることができますが、デバイスは、インストーラーがこれを選択できるようにすべてのヒントに一致する必要があります。

表13.3 サブフィールド

サブフィールド	説明
deviceName	**/dev/vda** や **/dev/disk/by-path/** などの Linux デバイス名を含む文字列。ストレージの場所への **/dev/disk/by-path/<device_path>** リンクを使用することを推奨します。ヒントは、実際の値と完全に一致する必要があります。
hctl	**0:0:0:0** などの SCSI バスアドレスを含む文字列。ヒントは、実際の値と完全に一致する必要があります。
model	ベンダー固有のデバイス識別子を含む文字列。ヒントは、実際の値のサブ文字列になります。
vendor	デバイスのベンダーまたは製造元の名前が含まれる文字列。ヒントは、実際の値のサブ文字列になります。
serialNumber	デバイスのシリアル番号を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
minSizeGigabytes	デバイスの最小サイズ (ギガバイト単位) を表す整数。
wwn	一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
rotational	デバイスがローテーションするディスクである (true) か、そうでないか (false) を示すブール値。

使用例

```
- name: master-0
  role: master
  rootDeviceHints:
    deviceName: "/dev/sda"
```

13.1.6. ネットワークの概要

最初の起動時にすべてのホストが支援サービスにチェックインできるように、**ランデブー IP** はエージェント ISO の生成時に認識されている必要があります。IP アドレスが動的ホスト設定プロトコル (DHCP) サーバーを使用して割り当てられている場合、**rendezvousIP** フィールドは、デプロイメントされたコントロールプレーンの一部になるホストの 1 つの IP アドレスに設定する必要があります。DHCP サーバーがない環境では、IP アドレスを静的に定義できます。

静的 IP アドレスに加えて、NMState 形式の任意のネットワーク設定を適用できます。これには、VLAN と NIC ボンディングが含まれます。

13.1.6.1. DHCP

推奨される方法: install-config.yaml および agent-config.yaml

rendezvousIP フィールドの値を指定する必要があります。**networkConfig** フィールドは空白のままにすることができます。

agent-config.yaml.file のサンプル

```
apiVersion: v1alpha1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 (1)
```

(1) ランデブーホストの IP アドレス。

13.1.6.2. 静的ネットワーキング

a. 推奨される方法: **install-config.yaml** および **agent-config.yaml**

agent-config.yaml.file のサンプル

```
cat > agent-config.yaml << EOF
apiVersion: v1alpha1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 (1)
hosts:
  - hostname: master-0
    interfaces:
      - name: eno1
        macAddress: 00:ef:44:21:e6:a5 (2)
    networkConfig:
      interfaces:
        - name: eno1
          type: ethernet
          state: up
          mac-address: 00:ef:44:21:e6:a5
          ipv4:
            enabled: true
            address:
```

```
            - ip: 192.168.111.80 3
              prefix-length: 23 4
          dhcp: false
      dns-resolver:
        config:
          server:
            - 192.168.111.1 5
      routes:
        config:
          - destination: 0.0.0.0/0
            next-hop-address: 192.168.111.1 6
            next-hop-interface: eno1
            table-id: 254
EOF
```

1. **rendezvousIP** フィールドに値が指定されていない場合、**networkConfig** フィールドで指定された静的 IP アドレスから 1 つのアドレスが選択されます。
2. 設定を適用するホストを決定するために使用される、ホスト上のインターフェイスの MAC アドレス。
3. ターゲットのベアメタルホストの静的 IP アドレス。
4. ターゲットのベアメタルホストの静的 IP アドレスのサブネット接頭辞。
5. ターゲットのベアメタルホストの DNS サーバー。
6. ノードトラフィックのネクストホップアドレス。これは、指定されたインターフェイスに設定される IP アドレスと同じサブネットにある必要があります。

b. オプションの方法: GitOps ZTP マニフェスト
GitOps ZTP カスタムリソースのオプションの方法は、6 つのカスタムリソースで構成されます。静的 IP は **nmstateconfig.yaml** で設定できます。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: NMStateConfig
metadata:
  name: master-0
  namespace: openshift-machine-api
  labels:
    cluster0-nmstate-label-name: cluster0-nmstate-label-value
spec:
  config:
    interfaces:
      - name: eth0
        type: ethernet
        state: up
        mac-address: 52:54:01:aa:aa:a1
        ipv4:
          enabled: true
          address:
            - ip: 192.168.122.2 1
              prefix-length: 23 2
          dhcp: false
```

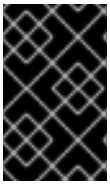
```
    dns-resolver:
      config:
        server:
          - 192.168.122.1 3
    routes:
      config:
        - destination: 0.0.0.0/0
          next-hop-address: 192.168.122.1 4
          next-hop-interface: eth0
          table-id: 254
interfaces:
  - name: eth0
    macAddress: 52:54:01:aa:aa:a1 5
```

1. ターゲットのベアメタルホストの静的 IP アドレス。
2. ターゲットのベアメタルホストの静的 IP アドレスのサブネット接頭辞。
3. ターゲットのベアメタルホストの DNS サーバー。
4. ノードトラフィックのネクストホップアドレス。これは、指定されたインターフェイスに設定される IP アドレスと同じサブネットにある必要があります。
5. 設定を適用するホストを決定するために使用される、ホスト上のインターフェイスの MAC アドレス。

ランデブー IP は、**config** フィールドで指定された静的 IP アドレスから選択されます。

13.1.7. プラットフォーム "none" オプションを使用するクラスターの要件

このセクションでは、プラットフォーム **none** オプションを使用するように設定されたエージェントベースの OpenShift Container Platform インストールの要件を説明します。



重要

仮想化またはクラウド環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

13.1.7.1. プラットフォーム "none" の DNS 要件

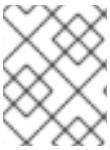
OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- コントロールプレーンおよびコンピュートマシン

逆引き DNS 解決は、Kubernetes API、コントロールプレーンマシン、およびコンピュートマシンにも必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用

されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。



注記

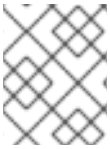
各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。

以下の DNS レコードは、プラットフォーム **none** オプションを使用する OpenShift Container Platform クラスターに必要であり、インストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表13.4 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>. <base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
コントロールプレーンマシン	**<master><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。

13.1.7.1.1. プラットフォーム "none" クラスターの DNS 設定例

このセクションでは、プラットフォーム **none** オプションを使用して OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

プラットフォーム "none" クラスターの DNS A レコード設定の例

次の例は、プラットフォーム **none** オプションを使用したクラスター内の名前解決のサンプル A レコードを示す BIND ゾーンファイルです。

例13.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
master0.ocp4.example.com. IN A 192.168.1.97 4
master1.ocp4.example.com. IN A 192.168.1.98 5
master2.ocp4.example.com. IN A 192.168.1.99 6
;
worker0.ocp4.example.com. IN A 192.168.1.11 7
worker1.ocp4.example.com. IN A 192.168.1.7 8
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 5 6 コントロールプレーンマシンの名前解決を提供します。

7 8 コンピュートマシンの名前解決を提供します。

- **プラットフォーム "none" クラスターの DNS PTR レコード設定の例**

次の BIND ゾーンファイルの例は、プラットフォーム **none** オプションを使用したクラスターでの逆名前解決のサンプル PTR レコードを示しています。

例13.2 逆引きレコードの DNS ゾーンデータベースの例

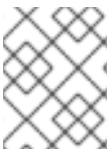
```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 3
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 4
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 5
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 6
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 7
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 4 5 コントロールプレーンマシンの逆引き DNS 解決を提供します。

6 7 コンピュートマシンの逆引き DNS 解決を提供します。

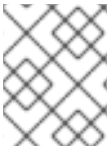


注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

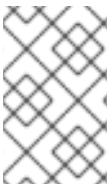
13.1.7.2. プラットフォーム "none" 負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

これらの要件は、プラットフォーム **none** オプションを使用する単一ノード OpenShift クラスターには適用されません。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

    - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
    - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

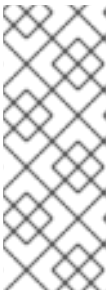
    重要

    API ロードバランサーのセッションの永続性は設定しないでください。

    ロードバランサーのフロントとバックの両方で以下のポートを設定します。

    表13.5 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	コントロールプレーンAPI サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	コントロールプレーン	X		マシン設定サーバー

    注記

    ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表13.6 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

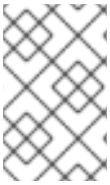
注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

13.1.7.2.1. プラットフォーム "none" クラスターのロードバランサー設定の例

このセクションでは、プラットフォーム **none** オプションを使用してクラスターのロードバランシング要件を満たす API とアプリケーションの Ingress ロードバランサー設定の例を示します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例13.3 API およびアプリケーション Ingress ロードバランサーの設定例

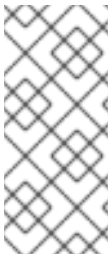
```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (2)
  bind *:22623
  mode tcp
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (3)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (4)
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1) ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

3 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

4 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

13.1.8. 例: ボンディングと VLAN インターフェイスノードのネットワーク設定

次の **agent-config.yaml** ファイルは、ボンディングおよび VLAN インターフェイスのマニフェストの例です。

```
apiVersion: v1alpha1
kind: AgentConfig
rendezvousIP: 10.10.10.14
hosts:
  - hostname: master0
    role: master
    interfaces:
     - name: enp0s4
       macAddress: 00:21:50:90:c0:10
     - name: enp0s5
       macAddress: 00:21:50:90:c0:20
    networkConfig:
      interfaces:
        - name: bond0.300 1
          type: vlan 2
          state: up
          vlan:
            base-iface: bond0
            id: 300
          ipv4:
            enabled: true
            address:
              - ip: 10.10.10.14
```

```
            prefix-length: 24
          dhcp: false
        - name: bond0 3
          type: bond 4
          state: up
          mac-address: 00:21:50:90:c0:10 5
          ipv4:
            enabled: false
          ipv6:
            enabled: false
          link-aggregation:
            mode: active-backup 6
            options:
              miimon: "150" 7
            port:
             - enp0s4
             - enp0s5
      dns-resolver: 8
        config:
          server:
            - 10.10.10.11
            - 10.10.10.12
      routes:
        config:
          - destination: 0.0.0.0/0
            next-hop-address: 10.10.10.10 9
            next-hop-interface: bond0.300 10
            table-id: 254
```

1 3 インターフェイスの名前。

2 インターフェイスのタイプ。以下の例では VLAN を作成します。

4 インターフェイスのタイプ。この例では、ボンドを作成します。

5 インターフェイスの MAC アドレス。

6 **mode** 属性は、ボンドモードを指定します。

7 MII リンクの監視頻度をミリ秒単位で指定します。この例では、ボンディングリンクを 150 ミリ秒ごとに検査します。

8 オプション:DNS サーバーの検索およびサーバー設定を指定します。

9 ノードトラフィックのネクストホップアドレス。これは、指定されたインターフェイスに設定される IP アドレスと同じサブネットにある必要があります。

10 ノードトラフィックのネクストホップインターフェイス。

13.1.9. 例: ボンディングと SR-IOV デュアル NIC ノードのネットワーク設定

重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

次の **agent-config.yaml** ファイルは、ボンドと SR-IOV インターフェイスを備えたデュアルポート NIC のマニフェストの例です。

```
apiVersion: v1alpha1
kind: AgentConfig
rendezvousIP: 10.10.10.14
hosts:
  - hostname: worker-1
    interfaces:
      - name: eno1
        macAddress: 0c:42:a1:55:f3:06
      - name: eno2
        macAddress: 0c:42:a1:55:f3:07
    networkConfig: 1
      interfaces: 2
        - name: eno1 3
          type: ethernet 4
          state: up
          mac-address: 0c:42:a1:55:f3:06
          ipv4:
            enabled: true
            dhcp: false 5
          ethernet:
            sr-iov:
              total-vfs: 2 6
          ipv6:
            enabled: false
        - name: sriov:eno1:0
          type: ethernet
          state: up 7
          ipv4:
            enabled: false 8
          ipv6:
            enabled: false
            dhcp: false
        - name: sriov:eno1:1
          type: ethernet
          state: down
        - name: eno2
          type: ethernet
          state: up
          mac-address: 0c:42:a1:55:f3:07
```

```
      ipv4:
        enabled: true
      ethernet:
        sr-iov:
          total-vfs: 2
      ipv6:
        enabled: false
    - name: sriov:eno2:0
      type: ethernet
      state: up
      ipv4:
        enabled: false
      ipv6:
        enabled: false
    - name: sriov:eno2:1
      type: ethernet
      state: down
    - name: bond0
      type: bond
      state: up
      min-tx-rate: 100 9
      max-tx-rate: 200 10
      link-aggregation:
        mode: active-backup 11
        options:
          primary: sriov:eno1:0 12
        port:
          - sriov:eno1:0
          - sriov:eno2:0
      ipv4:
        address:
          - ip: 10.19.16.57 13
            prefix-length: 23
        dhcp: false
        enabled: true
      ipv6:
        enabled: false
      dns-resolver:
        config:
          server:
            - 10.11.5.160
            - 10.2.70.215
      routes:
        config:
          - destination: 0.0.0.0/0
            next-hop-address: 10.19.17.254
            next-hop-interface: bond0 14
            table-id: 254
```

1. **networkConfig** フィールドには、ホストのネットワーク設定に関する情報が含まれ、サブフィールドには、**interfaces**、**dns-resolver**、および **routes** が含まれます。
2. **interfaces** フィールドは、ホスト用に定義されたネットワークインターフェイスの配列です。
3. インターフェイスの名前。

4 インターフェイスのタイプ。この例では、イーサネットインターフェイスを作成します。

5 厳密に必要ではない場合、物理機能 (PF) の DHCP を無効にするには、これを **false** に設定します。

6 これを、インスタンス化する SR-IOV 仮想機能 (VF) の数に設定します。

7 これを **up** に設定します。

8 ボンドに接続された VF の IPv4 アドレス指定を無効にするには、これを **false** に設定します。

9 VF の最小伝送速度 (Mbps) を設定します。このサンプル値は、100 Mbps のレートを設定します。

- この値は、最大伝送レート以下である必要があります。
- Intel NIC は **min-tx-rate** パラメーターをサポートしていません。詳細については、BZ#1772847 を参照してください。

10 VF の最大伝送速度 (Mbps) を設定します。このサンプル値は、200 Mbps のレートを設定します。

11 目的のボンディングモードを設定します。

12 ボンディングインターフェイスの優先ポートを設定します。プライマリーデバイスは、最初に使用されるボンディングインターフェイスであり、障害が発生しないかぎり、破棄されません。この設定が特に役立つのは、ボンディングインターフェイスの NIC の 1 つが高速なため、大規模な負荷に対応できる場合です。この設定は、ボンディングインターフェイスが **active-backup** モード (モード 1) および **balance-tlb** (モード 5) の場合のみに有効です。

13 ボンドインターフェイスの静的 IP アドレスを設定します。これはノードの IP アドレスです。

14 デフォルトルートのゲートウェイとして **bond0** を設定します。

関連情報

- ネットワークボンディングの設定

13.1.10. ベアメタルのサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

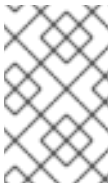
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- name: worker
  replicas: 0 3
controlPlane: 4
  name: master
  replicas: 1 5
metadata:
  name: sno-cluster 6
networking:
  clusterNetwork:
```

```
  - cidr: 10.128.0.0/14 7
    hostPrefix: 23 8
  networkType: OVNKubernetes 9
  serviceNetwork: 10
  - 172.30.0.0/16
platform:
  none: {} 11
fips: false 12
pullSecret: '{"auths": ...}' 13
sshKey: 'ssh-ed25519 AAAA...' 14
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 このパラメーターは、インストールプロセスをトリガーする前に、エージェントベースのインストールが検出を待機するコンピュートマシンの数を制御します。これは、生成された ISO で起動する必要があるコンピューティングマシンの数です。



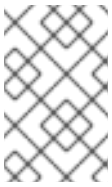
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

5 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

6 DNS レコードに指定したクラスター名。

7 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

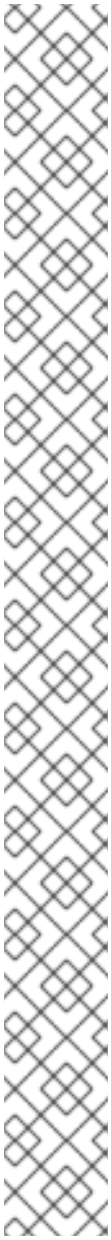
クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

8 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

9 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

10 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスに

11 単一ノードクラスターの場合は、プラットフォームを **none** に設定しなくてはなりません。マルチノードクラスターの場合は、プラットフォームを **vsphere**、**baremetal**、または **none** に設定できます。



注記

プラットフォームを **vsphere** または **baremetal** に設定すると、次の 3 つの方法でクラスターノードの IP アドレスエンドポイントを設定できます。

- IPv4
- IPv6
- IPv4 と IPv6 の並列 (デュアルスタック)

デュアルスタックネットワーキングの例

```
networking:
  clusterNetwork:
    - cidr: 172.21.0.0/16
      hostPrefix: 23
    - cidr: fd02::/48
      hostPrefix: 64
  machineNetwork:
    - cidr: 192.168.11.0/16
    - cidr: 2001:DB8::/32
  serviceNetwork:
    - 172.22.0.0/16
    - fd03::/112
  networkType: OVNKubernetes
platform:
  baremetal:
    apiVIPs:
    - 192.168.11.3
    - 2001:DB8::4
    ingressVIPs:
    - 192.168.11.4
    - 2001:DB8::5
```

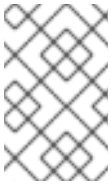
12 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

13 このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証で

14 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

13.1.11. agent.ISO の作成前の検証チェック

Agent-based Installer は、ISO が作成される前に、ユーザー定義の YAML ファイルに対して検証チェックを実行します。検証に成功すると、agent.ISO が作成されます。

install-config.yaml

- **baremetal**、**vsphere**、および **none** プラットフォームがサポートされています。
- **none** プラットフォームの場合、**networkType** パラメーターは **OVNKubernetes** である必要があります。
- **apiVIPs** および **ingressVIPs** パラメーターは、ベアメタルおよび vSphere プラットフォームに設定する必要があります。
- **agent-config.yaml** ファイルに相当するベアメタルプラットフォーム設定の一部のホスト固有フィールドは無視されます。これらのフィールドが設定されている場合、警告メッセージがログに記録されます。

agent-config.yaml

- 各インターフェイスには、定義された MAC アドレスが必要です。また、すべてのインターフェイスに異なる MAC アドレスが必要です。
- ホストごとに少なくとも 1 つのインターフェイスを定義する必要があります。
- World Wide Name (WWN) ベンダーエクステンションは、ルートデバイスのヒントではサポートされません。
- **host** オブジェクトの **role** パラメーターの値は **master** または **worker** のいずれかである必要があります。

13.1.11.1. ZTP マニフェスト

agent-cluster-install.yaml

- IPv6 の場合、**networkType** パラメーターでサポートされる唯一の値は **OVNKubernetes** です。**OpenshiftSDN** 値は、IPv4 にのみ使用できます。

cluster-image-set.yaml

- **ReleaseImage** パラメーターは、インストーラーで定義されるリリースと一致する必要があります。

13.1.12. 次のステップ

- Agent-based Installer を使用したクラスターのインストール

13.2. 切断されたインストールのミラーリングについて

切断されたインストールにミラーレジストリーを使用して、クラスターが外部コンテンツに対する組織の制御を満たすコンテナーイメージのみを使用するようにすることができます。ネットワークが切断された環境でプロビジョニングするインフラストラクチャーにクラスターをインストールする前に、必要なコンテナーイメージをその環境にミラーリングする必要があります。コンテナーイメージをミラーリングするには、ミラーリング用のレジストリーが必要です。

13.2.1. Agent-based Installer による非接続インストールのイメージのミラーリング

以下の手順のいずれかを使用して、OpenShift Container Platform イメージリポジトリーをミラーレジストリーにミラーリングできます。

- 非接続インストールのイメージのミラーリング
- oc-mirror プラグインを使用した非接続インストールのイメージのミラーリング

13.2.2. 切断されたレジストリーの OpenShift Container Platform イメージリポジトリーのミラーリング

Agent-based Installer による非接続インストールにミラーイメージを使用するには、**install-config.yaml** ファイルを変更する必要があります。

oc adm release mirror または **oc mirror** コマンドの出力を使用して、リリースイメージをミラーリングできます。これは、ミラーレジストリーの設定に使用したコマンドによって異なります。

次の例は、**oc adm release mirror** コマンドの出力を示しています。

```
$ oc adm release mirror
```

出力例

```
To use the new mirrored repository to install, add the following
section to the install-config.yaml:

imageContentSources:

mirrors:
virthost.ostest.test.metalkube.org:5000/localimages/local-release-image
source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
mirrors:
virthost.ostest.test.metalkube.org:5000/localimages/local-release-image
source: registry.ci.openshift.org/ocp/release
```

次の例は、oc-mirror プラグインによって生成された **imageContentSourcePolicy.yaml** ファイルの一部を示しています。このファイルは結果ディレクトリーにあります (例: **oc-mirror-workspace/results-1682697932/**)。

imageContentSourcePolicy.yaml ファイルの例

```
spec:
  repositoryDigestMirrors:
  - mirrors:
    - virthost.ostest.test.metalkube.org:5000/openshift/release
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
  - mirrors:
    - virthost.ostest.test.metalkube.org:5000/openshift/release-images
    source: quay.io/openshift-release-dev/ocp-release
```

13.2.2.1. ミラーリングされたイメージを使用するように Agent-based Installer を設定する

ミラーリングされたイメージを使用するように Agent-based Installer を設定するには、**oc adm release Mirror** コマンドまたは oc-mirror プラグインの出力を使用する必要があります。

手順

1. oc-mirror プラグインを使用してリリースイメージをミラーリングした場合:
   a. 結果ディレクトリーにある **imageContentSourcePolicy.yaml** を開きます (例: **oc-mirror-workspace/results-1682697932/**)。
   b. yaml ファイルの **repositoryDigestMirrors** セクションのテキストをコピーします。
2. **oc adm release Mirror** コマンドを使用してリリースイメージをミラーリングした場合:
   - コマンド出力の **imageContentSources** セクションのテキストをコピーします。
3. コピーしたテキストを **install-config.yaml** ファイルの **imageContentSources** フィールドに貼り付けます。
4. ミラーレジストリーに使用される証明書ファイルを yaml ファイルの **additionalTrustBundle** フィールドに追加します。

重要

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

install-config.yaml ファイルの例

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

5. GitOps ZTP マニフェストを使用している場合： **registries.conf** および **ca-bundle.crt** ファイルを **mirror** パスに追加して、エージェント ISO イメージにミラー設定を追加します。

注記

oc adm release Mirror コマンドまたは **oc Mirror** プラグインの出力から **registries.conf** ファイルを作成できます。**/etc/containers/registries.conf** ファイルの形式が変更されました。現在のバージョンはバージョン 2 で、TOML 形式です。

registries.conf ファイルの例

```
[[registry]]
location = "registry.ci.openshift.org/ocp/release" mirror-by-digest-only = true

[[registry.mirror]] location = "virthost.ostest.test.metalkube.org:5000/localimages/local-release-image"

[[registry]]
location = "quay.io/openshift-release-dev/ocp-v4.0-art-dev" mirror-by-digest-only = true

[[registry.mirror]] location = "virthost.ostest.test.metalkube.org:5000/localimages/local-release-image"
```

13.3. AGENT-BASED INSTALLER を使用した OPENSHIFT CONTAINER PLATFORM クラスターのインストール

以下の手順に従って、Agent-based Installer を使用して、OpenShift Container Platform クラスターをインストールします。

13.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ファイアウォールまたプロキシーを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

13.3.2. Agent-based Installer を使用した OpenShift Container Platform のインストール

以下の手順では、切断された環境でシングルノードの OpenShift Container Platform をデプロイします。これらの手順を基本として使用し、必要に応じて変更できます。

13.3.2.1. Agent-based Installer のダウンロード

この手順を使用して、インストールに必要なエージェントベースのインストーラーと CLI をダウンロードします。

手順

1. ログイン認証情報を使用して OpenShift Container Platform Web コンソールにログインします。
2. データセンター に移動します。

3. **Run Agent-based Installer locally** をクリックします。
4. **OpenShift インストーラー** と **コマンドラインインターフェイス** のオペレーティングシステムとアーキテクチャーを選択します。
5. **Download Installer** をクリックして、インストールプログラムをダウンロードして展開します。
6. **Download pull secret** または **Copy pull secret** をクリックして、プルシークレットをダウンロードまたはコピーできます。
7. **Download command-line tools** をクリックし、**openshift-install** バイナリーを **PATH** 上のディレクトリーに配置します。

13.3.2.2. エージェントベースのインストーラークラスターをインストールするためにサポートされているアーキテクチャーの確認

エージェントベースのインストーラーを使用して OpenShift Container Platform クラスターをインストールする前に、クラスターをインストールすることができるサポート対象のアーキテクチャーを検証できます。この手順はオプションです。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- インストールプログラムをダウンロードしている。

手順

1. OpenShift CLI (**oc**) にログインします。
2. 次のコマンドを実行して、リリースペイロードを確認します。

```
$ ./openshift-install version
```

出力例

```
./openshift-install 4.16.0
built from commit abc123def456
release image quay.io/openshift-release-dev/ocp-
release@sha256:123abc456def789ghi012jkl345mno678pqr901stu234vwx567yz0
release architecture amd64
```

multi ペイロードでリリースイメージを使用している場合、このコマンドの出力に表示される **リリースアーキテクチャー** がデフォルトのアーキテクチャーです。

3. ペイロードのアーキテクチャーを確認するには、次のコマンドを実行します。

```
$ oc adm release info <release_image> -o jsonpath="{ .metadata.metadata}" 1
```

1 &lt **;release_image>** をリリースイメージに置き換えます。例： **quay.io/openshift-release-dev/ocp-release@sha256:123abc456def789ghi012jkl345mno678pqr901stu234vwx567yz0**.

.example 出力先リリースイメージが **multi** ペイロードを使用する場合の出力：

```
{"release.openshift.io architecture":"multi"}
```

マルチ ペイロードでリリースイメージを使用している場合は、**arm64**、**amd64**、**s390x**、**ppc64le** など、異なるアーキテクチャーにクラスターをインストールできます。それ以外の場合は、**openshift-install version** コマンドの出力に表示される **リリースアーキテクチャー** にのみクラスターをインストールできます。

13.3.2.3. 優先設定入力の作成

この手順を使用して、エージェントイメージの作成に使用される優先設定入力を作成します。

手順

1. 以下のコマンドを実行して **nmstate** の依存関係をインストールします。

```
$ sudo dnf install /usr/bin/nmstatectl -y
```

2. PATH にあるディレクトリーに **openshift-install** バイナリーを配置します。

3. 次のコマンドを実行して、インストール設定を保存するディレクトリーを作成します。

```
$ mkdir ~/<directory_name>
```

注記

これは、エージェントベースのインストールで推奨される方法です。GitOps ZTP マニフェストの使用はオプションです。

4. **install-config.yaml** ファイルを作成します。

```
$ cat << EOF > ./my-cluster/install-config.yaml
apiVersion: v1
baseDomain: test.example.com
compute:
- architecture: amd64 (1)
  hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  replicas: 1
metadata:
  name: sno-cluster (2)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.0.0/16
```

```
  networkType: OVNKubernetes 3
  serviceNetwork:
  - 172.30.0.0/16
platform: 4
  none: {}
pullSecret: '<pull_secret>' 5
sshKey: '<ssh_pub_key>' 6
EOF
```

1 システムアーキテクチャーを指定します。有効な値は、**amd64**、**arm64**、**ppc64le**、および **s390x** です。

マルチ ペイロードでリリースイメージを使用している場合は、**arm64**、**amd64**、**s390x**、**ppc64le** など、異なるアーキテクチャーにクラスターをインストールできます。それ以外の場合は、**openshift-install version** コマンドの出力に表示される **リリースアーキテクチャー** にのみクラスターをインストールできます。詳細は、「エージェントベースのインストーラークラスターをインストールするためにサポートされているアーキテクチャーの検証」を参照してください。

2 必須。クラスター名を指定します。

3 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

4 プラットフォームを指定します。

注記

ベアメタルプラットフォームの場合、**agent-config.yaml** ファイル上での設定でオーバーライドされない限り、**install-config.yaml** ファイルのプラットフォームセクションでのホスト設定がデフォルトで使用されます。

5 プルシークレットを指定します。

6 SSH 公開鍵を指定します。

注記

プラットフォームを **vSphere** または **baremetal** に設定すると、クラスターノードの IP アドレスエンドポイントを次の 3 つの方法で設定できます。

- IPv4
- IPv6
- IPv4 と IPv6 の並列 (デュアルスタック)

IPv6 は、ベアメタルプラットフォームでのみサポートされます。

デュアルスタックネットワーキングの例

```
networking:
  clusterNetwork:
```

```
    - cidr: 172.21.0.0/16
      hostPrefix: 23
    - cidr: fd02::/48
      hostPrefix: 64
  machineNetwork:
    - cidr: 192.168.11.0/16
    - cidr: 2001:DB8::/32
  serviceNetwork:
    - 172.22.0.0/16
    - fd03::/112
  networkType: OVNKubernetes
platform:
  baremetal:
    apiVIPs:
    - 192.168.11.3
    - 2001:DB8::4
    ingressVIPs:
    - 192.168.11.4
    - 2001:DB8::5
```

5. **agent-config.yaml** ファイルを作成します。

```
$ cat > agent-config.yaml << EOF
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 (1)
hosts: (2)
  - hostname: master-0 (3)
    interfaces:
      - name: eno1
        macAddress: 00:ef:44:21:e6:a5
    rootDeviceHints: (4)
      deviceName: /dev/sdb
    networkConfig: (5)
      interfaces:
        - name: eno1
          type: ethernet
          state: up
          mac-address: 00:ef:44:21:e6:a5
          ipv4:
            enabled: true
            address:
              - ip: 192.168.111.80
                prefix-length: 23
            dhcp: false
      dns-resolver:
        config:
          server:
            - 192.168.111.1
      routes:
        config:
          - destination: 0.0.0.0/0
            next-hop-address: 192.168.111.2
```

```
        next-hop-interface: eno1
        table-id: 254
EOF
```

1. この IP アドレスは、ブートストラッププロセスを実行するノードや、**assisted-service** コンポーネントを実行するノードを判別するために使用されます。**networkConfig** パラメーターで少なくとも 1 つのホストの IP アドレスを指定しない場合は、ランデブー IP アドレスを指定する必要があります。このアドレスが指定されていない場合、指定されたホストの **networkConfig** から 1 つの IP アドレスが選択されます。
2. オプション: ホスト設定。定義されたホストの数は、**install-config.yaml** ファイルで定義されたホストの総数 (**compute.replicas** および **controlPlane.replicas** パラメーターの値の合計) を超えてはなりません。
3. オプション: 動的ホスト設定プロトコル (DHCP) または逆引き DNS ルックアップから取得したホスト名をオーバーライドします。各ホストには、これらの方法のいずれかによって提供される一意のホスト名が必要です。
4. 特定デバイスへの Red Hat Enterprise Linux CoreOS (RHCOS) イメージのプロビジョニングを有効にします。インストールプログラムは、検出順にデバイスを検査し、検出された値をヒントの値と比較します。ヒントの値と一致する最初に検出されたデバイスが使用されます。
5. オプション: ホストのネットワークインターフェイスを NMState 形式で設定します。

関連情報

- VMware vCenter のリージョンとゾーンの設定
- エージェントベースのインストーラークラスターをインストールするためにサポートされているアーキテクチャーの確認

13.3.2.4. オプション: 追加のマニフェストファイルの作成

追加のマニフェストを作成して、**install-config.yaml** ファイルと **agent-config.yaml** ファイルで使用可能な設定を超えてクラスターをさらに設定できます。

13.3.2.4.1. 追加のマニフェストを格納するディレクトリーの作成

追加のマニフェストを作成して、**install-config.yaml** ファイルおよび **agent-config.yaml** ファイルを超えてエージェントベースのインストールを設定する場合は、インストールディレクトリー内に **openshift** サブディレクトリーを作成する必要があります。追加のマシン設定は、すべてこのサブディレクトリーに配置する必要があります。

注記

最も一般的な追加マニフェストのタイプは **MachineConfig** オブジェクトです。エージェントベースのインストール中に追加できる **MachineConfig** オブジェクトの例については、"関連情報" セクションの "MachineConfig オブジェクトを使用してノードを設定する" を参照してください。

手順

- インストールホスト上で次のコマンドを実行して、インストールディレクトリー内に **openshift** サブディレクトリーを作成します。

```
$ mkdir <installation_directory>/openshift
```

関連情報

- MachineConfig オブジェクトを使用したノードの設定

13.3.2.4.2. ディスクパーティション設定

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

前提条件

- インストールディレクトリー内に **openshift** サブディレクトリーを作成している。

手順

1. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
```

```
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

2. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

13.3.2.5. オプション: ZTP マニフェストの使用

GitOps ゼロタッチプロビジョニング (ZTP) マニフェストを使用すると、**install-config.yaml** および **agent-config.yaml** ファイルの範囲を超える設定を使用してインストールを設定できます。

注記

GitOps ZTP マニフェストは、事前に **install-config.yaml** ファイルと **Agent-config.yaml** ファイルを設定したかどうかにかかわらず生成できます。**install-config.yaml** ファイルと **Agent-config.yaml** ファイルを設定することを選択した場合、設定は生成時に ZTP クラスターマニフェストにインポートされます。

前提条件

- 使用する **PATH** 上のディレクトリーに **openshift-install** バイナリーを配置している。
- オプション: **install-config.yaml** ファイルと **agent-config.yaml** ファイルを作成し、設定している。

手順

1. 次のコマンドを使用して、ZTP クラスターマニフェストを生成します。

```
$ openshift-install agent create cluster-manifests --dir <installation_directory>
```

重要

install-config.yaml ファイルと **agent-config.yaml** ファイルを作成した場合、それらのファイルは削除され、このコマンドで生成されたクラスターマニフェストに置き換えられます。

install-config.yaml ファイルと **agent-config.yaml** ファイルに対して行われた設定は、**openshift-install Agent create cluster-manifests** コマンドを実行すると ZTP クラスターマニフェストにインポートされます。

2. **cluster-manifests** ディレクトリーに移動します。

```
$ cd <installation_directory>/cluster-manifests
```

3. **cluster-manifests** ディレクトリー内のマニフェストファイルを設定します。サンプルファイルの詳細は、「サンプル GitOps ZTP カスタムリソース」セクションを参照してください。
4. 非接続クラスター: ZTP マニフェストを生成する前に **install-config.yaml** ファイルでミラー設定を定義しなかった場合は、次の手順を実行します。
   a. **mirror** ディレクトリーに移動します。

```
$ cd ../mirror
```

   b. **mirror** ディレクトリーにマニフェストファイルを設定します。

関連情報

- GitOps ZTP カスタムリソースのサンプル。
- GitOps ゼロタッチプロビジョニング (ZTP) の詳細は、ネットワーク遠端の課題 を参照してください。

13.3.2.6. オプション: ディスクの暗号化

Agent-based Installer を使用して OpenShift Container Platform をインストールするときに、この手順を使用してディスクまたはパーティションを暗号化します。

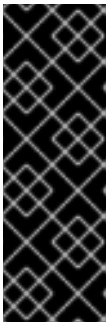
前提条件

- ZTP マニフェストを使用している場合を除き、**install-config.yaml** ファイルと **agent-config.yaml** ファイルが作成および設定されていること。
- 使用する **PATH** 上のディレクトリーに **openshift-install** バイナリーを配置している。

手順

1. 次のコマンドを使用して、ZTP クラスターマニフェストを生成します。

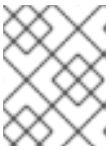
```
$ openshift-install agent create cluster-manifests --dir <installation_directory>
```



重要

install-config.yaml ファイルと **agent-config.yaml** ファイルを作成した場合、それらのファイルは削除され、このコマンドで生成されたクラスターマニフェストに置き換えられます。

install-config.yaml ファイルと **agent-config.yaml** ファイルに対して行われた設定は、**openshift-install Agent create cluster-manifests** コマンドを実行すると ZTP クラスターマニフェストにインポートされます。



注記

すでに ZTP マニフェストを生成している場合は、この手順をスキップしてください。

2. **cluster-manifests** ディレクトリーに移動します。

```
$ cd <installation_directory>/cluster-manifests
```

3. 次のセクションを **agent-cluster-install.yaml** ファイルに追加します。

```
diskEncryption:
  enableOn: all 1
  mode: tang 2
  tangServers: "server1": "http://tang-server-1.example.com:7500" 3
```

1. ディスク暗号化を有効にするノードを指定します。有効な値は、'none'、'all'、'master'、および 'worker' です。
2. 使用するディスク暗号化モードを指定します。有効な値は tpmv2 と tang です。
3. オプション: Tang を使用している場合は、Tang サーバーを指定します。

関連情報

- ディスクの暗号化について

13.3.2.7. エージェントイメージの作成と起動

この手順を使用して、マシンでエージェントイメージを起動します。

手順

1. 以下のコマンドを実行して agent イメージを作成します。

```
$ openshift-install --dir <install_directory> agent create image
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。マルチパス化は、デフォルトの **/etc/multipath.conf** 設定を使用して、agent.iSO イメージでデフォルトで有効になっています。

2. ベアメタルマシンで、**agent.x86_64.iso**、**agent.aarch64.iso**、または **agent.s390x.iso** イメージを起動します。

13.3.2.8. RHEL KVM を使用した IBM Z(R) エージェントの追加

RHEL KVM を使用して IBM Z® エージェントを手動で追加するには、次の手順を使用します。

注記

現在、IBM Z®(**s390x**)での ISO ブートのサポートは RHEL KVM でのみ利用できます。これにより、PXE または ISO ベースのインストールを柔軟に選択できます。z/VM を使用したインストールでは、PXE ブートのみがサポートされます。

手順

1. RHEL KVM マシンを起動します。
2. 仮想サーバーをデプロイするために、次のパラメーターを指定して **virt-install** コマンドを実行します。

```
$ virt-install
    --name <vm_name> \
    --autostart \
    --memory=<memory> \
    --cpu host \
    --vcpus=<vcpus> \
    --cdrom <agent.iso_image> \ 1
    --disk pool=default,size=<disk_pool_size> \
    --network network:default,mac=<mac_address> \
    --graphics none \
    --noautoconsole \
    --os-variant rhel9.0 \
    --wait=-1
```

1 the- **cdrom** パラメーターには、HTTP サーバーまたは HTTPS サーバー上の ISO イメージの場所を指定します。

13.3.2.9. 現在のインストールホストがリリースイメージをプルできることを確認する

エージェントイメージを起動し、ネットワークサービスがホストで利用可能になると、エージェントコンソールアプリケーションは、プルチェックを実行して、現在のホストがリリースイメージを取得できることを確認します。

一次プルチェックに合格した場合は、アプリケーションを終了して、インストールを続行できます。プルチェックが失敗した場合、アプリケーションは、TUI の **Additional checks** セクションに表示されるように、問題のトラブルシューティングに役立つ追加のチェックを実行します。追加のチェックの失敗は、プライマリープルチェックが成功するかぎり、必ずしも重大ではありません。

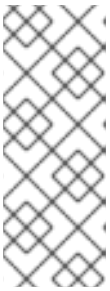
インストールが失敗する可能性があるホストネットワーク設定の問題がある場合は、コンソールアプリケーションを使用して、ネットワーク設定を調整できます。

重要

エージェントコンソールアプリケーションがホストネットワーク設定の問題を検出した場合は、ユーザーが手動でコンソールアプリケーションを停止し、続行する意思を示すまで、インストールワークフローは停止されます。

手順

1. エージェントコンソールアプリケーションが、設定されたリリースイメージをレジストリーからプルできるかどうかを確認するまで待ちます。
2. エージェントコンソールアプリケーションが、インストーラーの接続チェックに合格したことを示している場合は、プロンプトがタイムアウトになるまで待って、インストールを続行します。

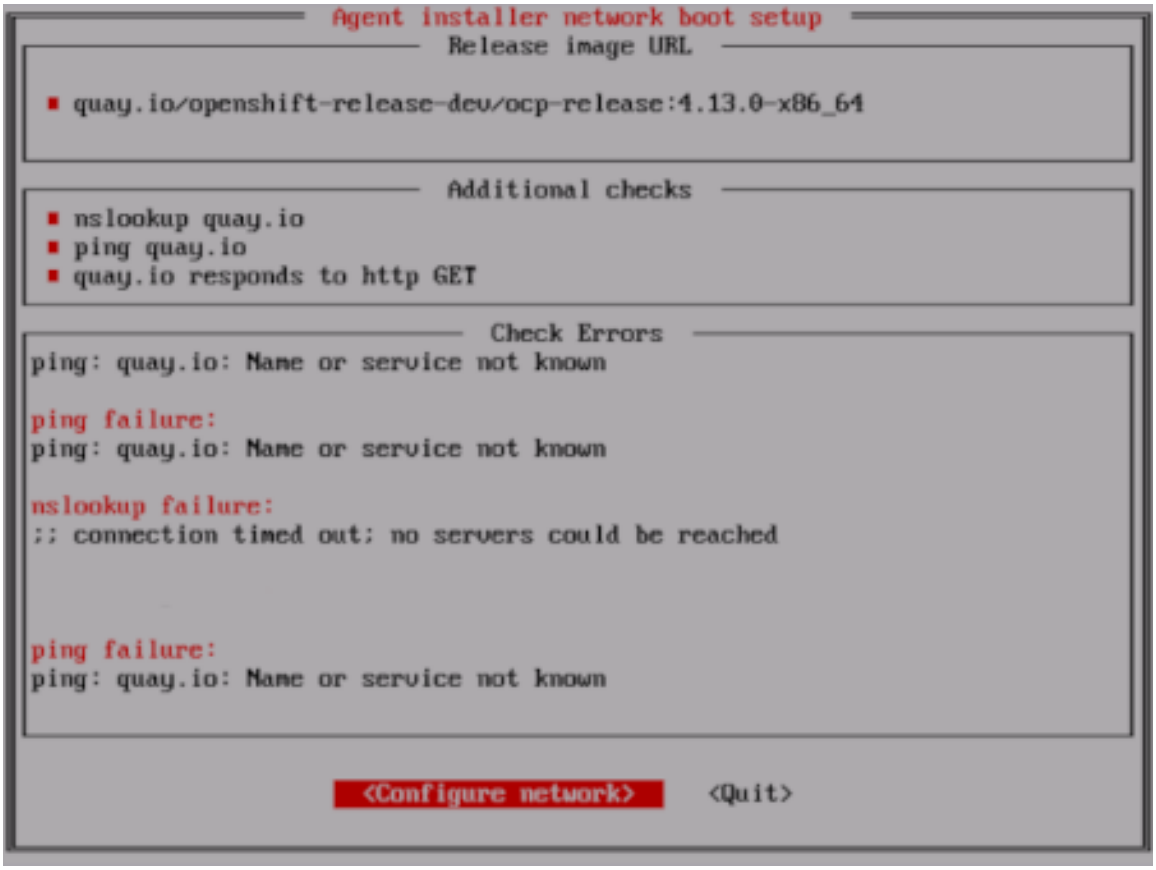


注記

接続チェックに合格した場合も、ネットワーク設定の表示または変更を選択できます。

ただし、タイムアウトせずに、エージェントコンソールアプリケーションとやりとりすることを選択した場合は、TUI を手動で終了して、インストールを続行する必要があります。

3. エージェントコンソールアプリケーションのチェックが失敗した場合は、**Release image URL** プルチェックの横に赤いアイコンが表示されます。ホストのネットワーク設定を再設定するには、次の手順を使用します。
    a. TUI の **Check Errors** セクションを読みます。このセクションには、失敗したチェックに固有のエラーメッセージが表示されます。



b. **Configure network** を選択して、NetworkManager TUI を起動します。

c. **Edit a connection** を選択し、再設定する接続を選択します。

d. 設定を編集し、**OK** を選択して、変更を保存します。

e. **Back** を選択して、NetworkManager TUI のメイン画面に戻ります。

f. **Activate a Connection** を選択します。

g. 再設定されたネットワークを選択して、非アクティブ化します。

h. 再設定されたネットワークを再度選択して、再アクティブ化します。

i. **Back** を選択し、**Quit** を選択して、エージェントコンソールアプリケーションに戻ります。

j. 新しいネットワーク設定を使用して、継続的なネットワークチェックが再開されるまで、5 秒間以上、待ちます。

k. **Release image URL** プルチェックが成功し、URL の横に緑色のアイコンが表示された場合は、**Quit** を選択して、エージェントコンソールアプリケーションを終了し、インストールを続行します。

13.3.2.10. インストールの進行状況の追跡と確認

次の手順を使用して、インストールの進行状況を追跡し、インストールが成功したことを確認します。

前提条件

- Kubernetes API サーバーの DNS レコードを設定している。

手順

1. オプション: ブートストラップホスト (ランデブーホスト) がいつ再起動するかを知るには、次のコマンドを実行します。

```
$ ./openshift-install --dir <install_directory> agent wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<install_directory>** には、エージェント ISO が生成されたディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
.....
.....
INFO Bootstrap configMap status is complete
INFO cluster bootstrap is complete
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. 進行状況を追跡し、インストールが成功したことを確認するには、次のコマンドを実行します。

```
$ openshift-install --dir <install_directory> agent wait-for install-complete 1
```

1 **<install_directory>** directory には、エージェント ISO が生成されたディレクトリーへのパスを指定します。

出力例

```
.....
.....
INFO Cluster is installed
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run
INFO     export KUBECONFIG=/home/core/installer/auth/kubeconfig
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.sno-cluster.test.example.com
```

注記

GitOps ZTP マニフェストのオプションの方法を使用している場合、次の 3 つの方法で **AgentClusterInstall.yaml** ファイルを介してクラスターノードの IP アドレスエンドポイントを設定できます。

- IPv4
- IPv6
- IPv4 と IPv6 の並列 (デュアルスタック)

IPv6 は、ベアメタルプラットフォームでのみサポートされます。

デュアルスタックネットワーキングの例

```
apiVIP: 192.168.11.3
ingressVIP: 192.168.11.4
clusterDeploymentRef:
  name: mycluster
imageSetRef:
  name: openshift-4.16
networking:
  clusterNetwork:
  - cidr: 172.21.0.0/16
    hostPrefix: 23
  - cidr: fd02::/48
    hostPrefix: 64
  machineNetwork:
  - cidr: 192.168.11.0/16
  - cidr: 2001:DB8::/32
  serviceNetwork:
  - 172.22.0.0/16
  - fd03::/112
  networkType: OVNKubernetes
```

関連情報

- デュアルスタックネットワークを使用したデプロイメント を参照してください。
- install-config yaml ファイルの設定 を参照してください。
- ベアメタル環境に 3 ノードクラスターをデプロイするための 3 ノードクラスターの設定 を参照してください。
- ルートデバイスヒントについて を参照してください。
- NMState 状態の例 を参照してください。

13.3.3. GitOps ZTP カスタムリソースのサンプル

オプション: GitOps ゼロタッチプロビジョニング (ZTP) カスタムリソース (CR) オブジェクトを使用して、Agent-based Installer で OpenShift Container Platform クラスターをインストールできます。

以下の GitOps ZTP カスタムリソースをカスタマイズして、OpenShift Container Platform クラスターの詳細を指定できます。次のサンプル GitOps ZTP カスタムリソースは、単一ノードクラスター用です。

agent-cluster-install.yaml

```
apiVersion: extensions.hive.openshift.io/v1beta1
kind: AgentClusterInstall
metadata:
  name: test-agent-cluster-install
  namespace: cluster0
spec:
  clusterDeploymentRef:
    name: ostest
  imageSetRef:
    name: openshift-4.16
  networking:
    clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
    serviceNetwork:
    - 172.30.0.0/16
  provisionRequirements:
    controlPlaneAgents: 1
    workerAgents: 0
  sshPublicKey: <YOUR_SSH_PUBLIC_KEY>
```

cluster-deployment.yaml

```
apiVersion: hive.openshift.io/v1
kind: ClusterDeployment
metadata:
  name: ostest
  namespace: cluster0
spec:
  baseDomain: test.metalkube.org
  clusterInstallRef:
    group: extensions.hive.openshift.io
    kind: AgentClusterInstall
    name: test-agent-cluster-install
    version: v1beta1
  clusterName: ostest
  controlPlaneConfig:
    servingCertificates: {}
  platform:
    agentBareMetal:
      agentSelector:
        matchLabels:
          bla: aaa
  pullSecretRef:
    name: pull-secret
```

cluster-image-set.yaml

```
apiVersion: hive.openshift.io/v1
```

```
kind: ClusterImageSet
metadata:
  name: openshift-4.16
spec:
  releaseImage: registry.ci.openshift.org/ocp/release:4.16.0-0.nightly-2022-06-06-025509
```

infra-env.yaml

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: myinfraenv
  namespace: cluster0
spec:
  clusterRef:
    name: ostest
    namespace: cluster0
  cpuArchitecture: aarch64
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: <YOUR_SSH_PUBLIC_KEY>
  nmStateConfigLabelSelector:
    matchLabels:
      cluster0-nmstate-label-name: cluster0-nmstate-label-value
```

nmstateconfig.yaml

```
apiVersion: agent-install.openshift.io/v1beta1
kind: NMStateConfig
metadata:
  name: master-0
  namespace: openshift-machine-api
  labels:
    cluster0-nmstate-label-name: cluster0-nmstate-label-value
spec:
  config:
    interfaces:
      - name: eth0
        type: ethernet
        state: up
        mac-address: 52:54:01:aa:aa:a1
        ipv4:
          enabled: true
          address:
            - ip: 192.168.122.2
              prefix-length: 23
          dhcp: false
    dns-resolver:
      config:
        server:
          - 192.168.122.1
    routes:
      config:
        - destination: 0.0.0.0/0
          next-hop-address: 192.168.122.1
```

```
        next-hop-interface: eth0
        table-id: 254
  interfaces:
    - name: "eth0"
      macAddress: 52:54:01:aa:aa:a1
```

pull-secret.yaml

```
apiVersion: v1
kind: Secret
type: kubernetes.io/dockerconfigjson
metadata:
  name: pull-secret
  namespace: cluster0
stringData:
  .dockerconfigjson: 'YOUR_PULL_SECRET'
```

関連情報

- GitOps ゼロタッチプロビジョニング (ZTP) の詳細は、ネットワーク遠端の課題 を参照してください。

13.3.4. 失敗したエージェントベースのインストールからログデータを収集する

次の手順を使用して、失敗したエージェントベースのインストールに関するログデータを収集し、サポートケースで提供できるよう備えます。

前提条件

- Kubernetes API サーバーの DNS レコードを設定している。

手順

1. 次のコマンドを実行し、出力を収集します。

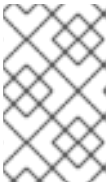
```
$ ./openshift-install --dir <install_directory> agent wait-for bootstrap-complete --log-level=debug
```

エラーメッセージの例

```
...
ERROR Bootstrap failed to complete: : bootstrap process timed out: context deadline exceeded
```

2. 直前のコマンド出力で障害の発生が示された場合、またはブートストラップが進行していない場合は、次のコマンドを実行してランデブーホストに接続し、出力を収集します。

```
$ ssh core@<node-ip> agent-gather -O >agent-gather.tar.xz
```



注記

Red Hat サポートは、ランデブーホストから収集したデータを使用してほとんどの問題を診断できますが、一部のホストが登録できない場合はすべてのホストからこのデータを収集すると役立ちます。

3. ブートストラップが完了し、クラスターノードが再起動したら、次のコマンドを実行して出力を収集します。

```
$ ./openshift-install --dir <install_directory> agent wait-for install-complete --log-level=debug
```

4. 前のコマンドの出力が失敗を示している場合は、次の手順を実行します。

   a. 次のコマンドを実行して、**kubeconfig** ファイルを環境にエクスポートします。

   ```
   $ export KUBECONFIG=<install_directory>/auth/kubeconfig
   ```

   b. デバッグ用の情報を収集するには、次のコマンドを実行します。

   ```
   $ oc adm must-gather
   ```

   c. 次のコマンドを実行して、作業ディレクトリーに作成した **must-gather** ディレクトリーから圧縮ファイルを作成します。

   ```
   $ tar cvaf must-gather.tar.gz <must_gather_directory>
   ```

5. **/auth** サブディレクトリーを除き、デプロイメント中に使用したインストールディレクトリーを Red Hat カスタマーポータル のサポートケースに添付します。

6. この手順で収集した他のすべてのデータをサポートケースに添付してください。

13.4. OPENSHIFT CONTAINER PLATFORM 用の PXE アセットの準備

Agent-based Installer を使用して OpenShift Container Platform クラスターを PXE ブートするために必要なアセットを作成するには、以下の手順を使用します。

これらの手順で作成したアセットは、単一ノードの OpenShift Container Platform インストールをデプロイします。これらの手順を基礎として使用し、要件に応じて設定を変更できます。

13.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。

13.4.2. Agent-based Installer のダウンロード

この手順を使用して、インストールに必要なエージェントベースのインストーラーと CLI をダウンロードします。

手順

1. ログイン認証情報を使用して OpenShift Container Platform Web コンソールにログインします。

2. データセンター に移動します。
3. **Run Agent-based Installer locally** をクリックします。
4. **OpenShift インストーラー** と **コマンドラインインターフェイス** のオペレーティングシステムとアーキテクチャーを選択します。
5. **Download Installer** をクリックして、インストールプログラムをダウンロードして展開します。
6. **Download pull secret** または **Copy pull secret** をクリックして、プルシークレットをダウンロードまたはコピーできます。
7. **Download command-line tools** をクリックし、**openshift-install** バイナリーを **PATH** 上のディレクトリーに配置します。

13.4.3. 優先設定入力の作成

この手順を使用して、PXE ファイルの作成に使用される優先設定入力を作成します。

手順

1. 以下のコマンドを実行して **nmstate** の依存関係をインストールします。

```
$ sudo dnf install /usr/bin/nmstatectl -y
```

2. PATH にあるディレクトリーに **openshift-install** バイナリーを配置します。
3. 次のコマンドを実行して、インストール設定を保存するディレクトリーを作成します。

```
$ mkdir ~/<directory_name>
```

注記

これは、エージェントベースのインストールで推奨される方法です。GitOps ZTP マニフェストの使用はオプションです。

4. **install-config.yaml** ファイルを作成します。

```
$ cat << EOF > ./my-cluster/install-config.yaml
apiVersion: v1
baseDomain: test.example.com
compute:
- architecture: amd64 (1)
  hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  replicas: 1
metadata:
  name: sno-cluster (2)
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.0.0/16
  networkType: OVNKubernetes 3
  serviceNetwork:
  - 172.30.0.0/16
platform: 4
  none: {}
pullSecret: '<pull_secret>' 5
sshKey: '<ssh_pub_key>' 6
EOF
```

1 システムアーキテクチャーを指定します。有効な値は、**amd64**、**arm64**、**ppc64le**、および **s390x** です。

マルチ ペイロードでリリースイメージを使用している場合は、**arm64**、**amd64**、**s390x**、**ppc64le** など、異なるアーキテクチャーにクラスターをインストールできます。それ以外の場合は、**openshift-install version** コマンドの出力に表示される **リリースアーキテクチャー** にのみクラスターをインストールできます。詳細は、「エージェントベースのインストーラークラスターをインストールするためにサポートされているアーキテクチャーの検証」を参照してください。

2 必須。クラスター名を指定します。

3 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

4 プラットフォームを指定します。

注記

ベアメタルプラットフォームの場合、**agent-config.yaml** ファイル上での設定でオーバーライドされない限り、**install-config.yaml** ファイルのプラットフォームセクションでのホスト設定がデフォルトで使用されます。

5 プルシークレットを指定します。

6 SSH 公開鍵を指定します。

注記

プラットフォームを **vSphere** または **baremetal** に設定すると、クラスターノードの IP アドレスエンドポイントを次の 3 つの方法で設定できます。

- IPv4
- IPv6
- IPv4 と IPv6 の並列 (デュアルスタック)

IPv6 は、ベアメタルプラットフォームでのみサポートされます。

デュアルスタックネットワーキングの例

```
networking:
  clusterNetwork:
    - cidr: 172.21.0.0/16
      hostPrefix: 23
    - cidr: fd02::/48
      hostPrefix: 64
  machineNetwork:
    - cidr: 192.168.11.0/16
    - cidr: 2001:DB8::/32
  serviceNetwork:
    - 172.22.0.0/16
    - fd03::/112
  networkType: OVNKubernetes
platform:
  baremetal:
    apiVIPs:
    - 192.168.11.3
    - 2001:DB8::4
    ingressVIPs:
    - 192.168.11.4
    - 2001:DB8::5
```

5. **agent-config.yaml** ファイルを作成します。

```
$ cat > agent-config.yaml << EOF
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 (1)
hosts: (2)
  - hostname: master-0 (3)
    interfaces:
      - name: eno1
        macAddress: 00:ef:44:21:e6:a5
    rootDeviceHints: (4)
      deviceName: /dev/sdb
    networkConfig: (5)
      interfaces:
        - name: eno1
          type: ethernet
          state: up
          mac-address: 00:ef:44:21:e6:a5
          ipv4:
            enabled: true
            address:
              - ip: 192.168.111.80
                prefix-length: 23
            dhcp: false
      dns-resolver:
        config:
          server:
            - 192.168.111.1
```

```
    routes:
      config:
        - destination: 0.0.0.0/0
          next-hop-address: 192.168.111.2
          next-hop-interface: eno1
          table-id: 254
EOF
```

1. この IP アドレスは、ブートストラッププロセスを実行するノードや、**assisted-service** コンポーネントを実行するノードを判別するために使用されます。**networkConfig** パラメーターで少なくとも 1 つのホストの IP アドレスを指定しない場合は、ランデブー IP アドレスを指定する必要があります。このアドレスが指定されていない場合、指定されたホストの **networkConfig** から 1 つの IP アドレスが選択されます。
2. オプション: ホスト設定。定義されたホストの数は、**install-config.yaml** ファイルで定義されたホストの総数 (**compute.replicas** および **controlPlane.replicas** パラメーターの値の合計) を超えてはなりません。
3. オプション: 動的ホスト設定プロトコル (DHCP) または逆引き DNS ルックアップから取得したホスト名をオーバーライドします。各ホストには、これらの方法のいずれかによって提供される一意のホスト名が必要です。
4. 特定デバイスへの Red Hat Enterprise Linux CoreOS (RHCOS) イメージのプロビジョニングを有効にします。インストールプログラムは、検出順にデバイスを検査し、検出された値をヒントの値と比較します。ヒントの値と一致する最初に検出されたデバイスが使用されます。
5. オプション: ホストのネットワークインターフェイスを NMState 形式で設定します。

6. オプション: iPXE スクリプトを作成するには、**bootArtifactsBaseURL** を **agent-config.yaml** ファイルに追加します。

```
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80
bootArtifactsBaseURL: <asset_server_URL>
```

<asset_server_URL> は、PXE アセットをアップロードするサーバーの URL です。

関連情報

- デュアルスタックネットワークを使用したデプロイメント。
- install-config.yaml ファイルを設定します。
- ベアメタル環境に 3 ノードクラスターをデプロイするための 3 ノードクラスターの設定 を参照してください。
- ルートデバイスヒントについて。
- NMState 状態の例。
- オプション: 追加のマニフェストファイルの作成

13.4.4. PXE アセットの作成

次の手順を使用して、PXE インフラストラクチャーに実装するアセットとオプションのスクリプトを作成します。

手順

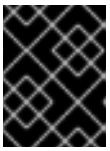
1. 次のコマンドを実行して、PXE アセットを作成します。

```
$ openshift-install agent create pxe-files
```

生成された PXE アセットと任意の iPXE スクリプトは、**boot-artifacts** ディレクトリーにあります。

PXE アセットとオプションの iPXE スクリプトを含むファイルシステムの例

```
boot-artifacts
    ├─ agent.x86_64-initrd.img
    ├─ agent.x86_64.ipxe
    ├─ agent.x86_64-rootfs.img
    └─ agent.x86_64-vmlinuz
```



重要

boot-artifacts ディレクトリーの内容は、指定したアーキテクチャーによって異なります。

注記

Red Hat Enterprise Linux CoreOS (RHCOS) はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。マルチパス化は、デフォルトの **/etc/multipath.conf** 設定を使用して、agent.iSO イメージでデフォルトで有効になっています。

2. PXE アセットと任意のスクリプトをインフラストラクチャーにアップロードし、ブートプロセス中にアクセスできるようにします。

注記

iPXE スクリプトを生成した場合、アセットの場所は、**agent-config.yaml** ファイルに追加した **bootArtifactsBaseURL** と一致する必要があります。

13.4.5. IBM Z エージェントの手動追加

PXE アセットを作成した後、IBM Z® エージェントを追加できます。

IBM Z® 環境に応じて、以下の方法から選択できます。

- z/VM を使用した IBM Z® エージェントの追加
- RHEL KVM を使用した IBM Z® エージェントの追加

13.4.5.1. z/VM を使用した IBM Z エージェントの追加

z/VM を使用して IBM Z® エージェントを手動で追加するには、次の手順を使用します。

手順

1. z/VM ゲストのパラメーターファイルを作成します。

   パラメーターファイルの例

   ```
   rd.neednet=1 \
   console=ttysclp0 \
   coreos.live.rootfs_url=<rootfs_url> \ 1
   ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \ 2
   zfcp.allow_lun_scan=0 \ 3
   rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
   rd.dasd=0.0.4411 \ 4
   rd.zfcp=0.0.8001,0x50050763040051e3,0x4000406300000000 \ 5
   random.trust_cpu=on rd.luks.options=discard \
   ignition.firstboot ignition.platform.id=metal \
   console=tty1 console=ttyS1,115200n8 \
   coreos.inst.persistent-kargs="console=tty1 console=ttyS1,115200n8"
   ```

   1. **coreos.live.rootfs_url** アーティファクトには、起動する **kernel** と **initramfs** に合った **rootfs** アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

   2. **ip=** パラメーターには、DHCP を使用して自動的に IP アドレスを割り当てるか、「z/VM を使用したクラスターの IBM Z® および IBM® LinuxONE へのインストール」の説明に従って手動で割り当てます。

   3. デフォルトは **1** です。OSA ネットワークアダプターを使用する場合は、このエントリーを省略してください。

   4. DASD タイプのディスクにインストールする場合は、**rd.dasd** を使用して、Red Hat Enterprise Linux CoreOS (RHCOS) をインストールする DASD を指定します。FCP タイプのディスクの場合は、このエントリーを省略します。

   5. FCP タイプのディスクにインストールする場合は、**rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して、RHCOS がインストールされる FCP ディスクを指定します。DASD タイプのディスクの場合は、このエントリーを省略します。

   その他のパラメーターはすべて変更しません。

2. z/VM ゲスト仮想マシンの仮想リーダーに対して、**kernel.img**、**generic.parm**、および **initrd.img** ファイルの punch を実行します。
   詳細は、IBM ドキュメントの PUNCH を参照してください。

   ヒント

   CP PUNCH コマンドを使用するか、Linux を使用している場合は **vmur** コマンドを使用して、2 つの z/VM ゲスト仮想マシン間でファイルを転送できます。

3. ブートストラップマシン上の会話型モニターシステム (CMS) にログインします。

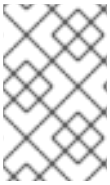
4. 次のコマンドを実行して、リーダーからブートストラップマシンを IPL します。

```
$ ipl c
```

詳細は、IBM ドキュメントの IPL を参照してください。

13.4.5.2. RHEL KVM を使用した IBM Z(R) エージェントの追加

RHEL KVM を使用して IBM Z® エージェントを手動で追加するには、次の手順を使用します。



注記

現在、IBM Z®(**s390x**)での ISO ブートのサポートは RHEL KVM でのみ利用できます。これにより、PXE または ISO ベースのインストールを柔軟に選択できます。z/VM を使用したインストールでは、PXE ブートのみがサポートされます。

手順

1. RHEL KVM マシンを起動します。
2. 仮想サーバーをデプロイするために、次のパラメーターを指定して **virt-install** コマンドを実行します。

```
$ virt-install \
  --name <vm_name> \
  --autostart \
  --ram=16384 \
  --cpu host \
  --vcpus=8 \
  --location <path_to_kernel_initrd_image>,kernel=kernel.img,initrd=initrd.img \ 1
  --disk <qcow_image_path> \
  --network network:macvtap ,mac=<mac_address> \
  --graphics none \
  --noautoconsole \
  --wait=-1 \
  --extra-args "rd.neednet=1 nameserver=<nameserver>" \
  --extra-args "ip=<IP>::<nameserver>::<hostname>:enc1:none" \
  --extra-args "coreos.live.rootfs_url=http://<http_server>:8080/agent.s390x-rootfs.img" \
  --extra-args "random.trust_cpu=on rd.luks.options=discard" \
  --extra-args "ignition.firstboot ignition.platform.id=metal" \
  --extra-args "console=tty1 console=ttyS1,115200n8" \
  --extra-args "coreos.inst.persistent-kargs=console=tty1 console=ttyS1,115200n8" \
  --osinfo detect=on,require=off
```

1 **--location** パラメーターには、HTTP または HTTPS サーバー上の kernel/initrd の場所を指定します。

13.4.6. 関連情報

- Agent-based Installer で使用できるその他の設定は、Agent-based Installer を使用した OpenShift Container Platform クラスターのインストール を参照してください。

13.5. KUBERNETES OPERATOR のマルチクラスターエンジン用の AGENT ベースのインストール済みクラスターの準備

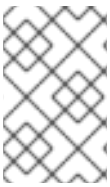
マルチクラスターエンジン Operator をインストールし、エージェントベースの OpenShift Container Platform インストーラーを使用してハブクラスターをデプロイできます。次の手順は部分的に自動化されており、最初のクラスターがデプロイメントされた後に手動の手順が必要です。

13.5.1. 前提条件

- 次のドキュメントを読んでいる。
  - マルチクラスターエンジン Operator を使用したクラスターライフサイクルについて
  - ローカルボリュームを使用した永続ストレージ
  - GitOps ZTP を使用したネットワーク遠端でのクラスタープロビジョニング
  - Agent-based Installer を使用したインストールの準備
  - 非接続インストールミラーリングについて
- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。
- 切断された環境にインストールする場合は、切断されたインストールのミラーリング用に設定されたローカルミラーレジストリーが必要です。

13.5.2. 切断中の Kubernetes Operator のマルチクラスターエンジン用のエージェントベースのクラスターデプロイの準備

切断された環境で、必要な OpenShift Container Platform コンテナーイメージ、マルチクラスターエンジン Operator、および Local Storage Operator (LSO) をローカルミラーレジストリーにミラーリングできます。ミラーレジストリーのローカル DNS ホスト名とポートを必ず書き留めておいてください。



注記

OpenShift Container Platform イメージリポジトリーをミラーレジストリーにミラーリングするには、**oc adm release image** または **oc mirror** コマンドを使用できます。この手順では、**oc mirror** コマンドを例として使用します。

手順

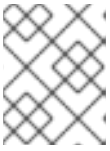
1. **<assets_directory>** フォルダーを作成して、有効な **install-config.yaml** および **agent-config.yaml** ファイルを含めます。このディレクトリーは、すべてのアセットを格納するために使用されます。
2. OpenShift Container Platform イメージリポジトリー、マルチクラスターエンジン、および LSO をミラーリングするには、以下の設定で **ImageSetConfiguration.yaml** ファイルを作成します。

   ImageSetConfiguration.yaml の例

   ```
   kind: ImageSetConfiguration
   apiVersion: mirror.openshift.io/v1alpha2
   ```

```
  archiveSize: 4 1
  storageConfig: 2
    imageURL: <your-local-registry-dns-name>:<your-local-registry-port>/mirror/oc-mirror-
metadata 3
    skipTLS: true
  mirror:
    platform:
      architectures:
        - "amd64"
      channels:
        - name: stable-4.16 4
          type: ocp
    additionalImages:
      - name: registry.redhat.io/ubi9/ubi:latest
    operators:
      - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16 5
        packages: 6
          - name: multicluster-engine 7
          - name: local-storage-operator 8
```

1. イメージセット内の各ファイルの最大サイズを GiB 単位で指定します。
2. イメージセットのメタデータを受け取るバックエンドの場所を設定します。この場所は、レジストリーまたはローカルディレクトリーにすることができます。**storageConfig**値を指定する必要があります。
3. ストレージバックエンドのレジストリー URL を設定します。
4. インストールするバージョンの OpenShift Container Platform イメージを含むチャネルを設定します。
5. インストールする OpenShift Container Platform イメージを含む Operator カタログを設定します。
6. イメージセットに含める特定の Operator パッケージとチャネルのみを指定します。カタログ内のすべてのパッケージを取得するには、このフィールドを削除してください。
7. マルチクラスターエンジンのパッケージとチャネル。
8. LSO パッケージとチャネル。



注記

このファイルは、コンテンツをミラーリングするときに **oc mirror** コマンドで必要になります。

3. 特定の OpenShift Container Platform イメージリポジトリー、マルチクラスターエンジン、および LSO をミラーリングするには、以下のコマンドを実行します。

```
$ oc mirror --dest-skip-tls --config ocp-mce-imageset.yaml docker://<your-local-registry-dns-
name>:<your-local-registry-port>
```

4. **install-config.yaml** ファイルのレジストリーと証明書を更新します。

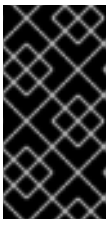
Example imageContentSources.yaml

```
imageContentSources:
  - source: "quay.io/openshift-release-dev/ocp-release"
    mirrors:
      - "<your-local-registry-dns-name>:<your-local-registry-port>/openshift/release-images"
  - source: "quay.io/openshift-release-dev/ocp-v4.0-art-dev"
    mirrors:
      - "<your-local-registry-dns-name>:<your-local-registry-port>/openshift/release"
  - source: "registry.redhat.io/ubi9"
    mirrors:
      - "<your-local-registry-dns-name>:<your-local-registry-port>/ubi9"
  - source: "registry.redhat.io/multicluster-engine"
    mirrors:
      - "<your-local-registry-dns-name>:<your-local-registry-port>/multicluster-engine"
  - source: "registry.redhat.io/rhel8"
    mirrors:
      - "<your-local-registry-dns-name>:<your-local-registry-port>/rhel8"
  - source: "registry.redhat.io/redhat"
    mirrors:
      - "<your-local-registry-dns-name>:<your-local-registry-port>/redhat"
```

さらに、証明書が **install-config.yaml** の **additionalTrustBundle** フィールドに存在することを確認してください。

install-config.yaml の例

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  zzzzzzzzzzz
  -----END CERTIFICATE-------
```



重要

oc mirror コマンドは、いくつかの出力を含む **oc-mirror-workspace** というフォルダーを作成します。これには、OpenShift Container Platform および選択した Operator に必要なすべてのミラーを識別する **imageContentSourcePolicy.yaml** ファイルが含まれます。

5. 次のコマンドを実行して、クラスターマニフェストを生成します。

```
$ openshift-install agent create cluster-manifests
```

このコマンドは、クラスターマニフェストフォルダーを更新して、ミラー設定を含む**mirror** フォルダーを含めます。

13.5.3. 接続中の Kubernetes Operator のマルチクラスターエンジン用のエージェントベースのクラスターデプロイメントの準備

マルチクラスターエンジン Operator である Local Storage Operator (LSO) に必要なマニフェストを作成し、エージェントベースの OpenShift Container Platform クラスターをハブクラスターとしてデプロイします。

手順

手順

1. **<assets_directory>** フォルダーに **openshift** という名前のサブフォルダーを作成します。このサブフォルダーは、デプロイされたクラスターをさらにカスタマイズするためにインストール中に適用される追加のマニフェストを格納するために使用されます。**<assets_directory>** フォルダーには、**install-config.yaml** および **agent-config.yaml** ファイルを含むすべてのアセットが含まれています。

注記

インストーラーは、追加のマニフェストを検証しません。

2. マルチクラスターエンジンの場合、以下のマニフェストを作成し、それらを **<assets_directory>/openshift** フォルダーに保存します。

例 mce_namespace.yaml

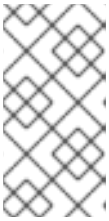
```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    openshift.io/cluster-monitoring: "true"
  name: multicluster-engine
```

例 mce_operatorgroup.yaml

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: multicluster-engine-operatorgroup
  namespace: multicluster-engine
spec:
  targetNamespaces:
  - multicluster-engine
```

例 mce_subscription.yaml

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: multicluster-engine
  namespace: multicluster-engine
spec:
  channel: "stable-2.3"
  name: multicluster-engine
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```



注記

アシステッドインストーラー(AI) を使用して、Red Hat Advanced Cluster Management (RHACM) で分散ユニット (DU) を大規模にインストールできます。これらの分散ユニットは、ハブクラスターで有効にする必要があります。AI サービスには、手動で作成される永続ボリューム (PV) が必要です。

3. AI サービスの場合、以下のマニフェストを作成し、それらを **<assets_directory>/openshift** フォルダーに保存します。

Example lso_namespace.yaml

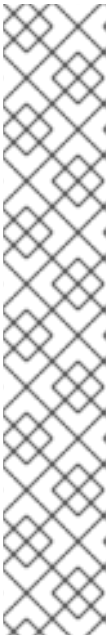
```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    openshift.io/cluster-monitoring: "true"
  name: openshift-local-storage
```

Example lso_operatorgroup.yaml

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: local-operator-group
  namespace: openshift-local-storage
spec:
  targetNamespaces:
    - openshift-local-storage
```

Example lso_subscription.yaml

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: local-storage-operator
  namespace: openshift-local-storage
spec:
  installPlanApproval: Automatic
  name: local-storage-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```



注記

すべてのマニフェストを作成した後、ファイルシステムは次のように表示される必要があります。

例: ファイルシステム

```
<assets_directory>
    ├─ install-config.yaml
    ├─ agent-config.yaml
    └─ /openshift
        ├─ mce_namespace.yaml
        ├─ mce_operatorgroup.yaml
        ├─ mce_subscription.yaml
        ├─ lso_namespace.yaml
        ├─ lso_operatorgroup.yaml
        └─ lso_subscription.yaml
```

4. 次のコマンドを実行して、エージェント ISO イメージを作成します。

```
$ openshift-install agent create image --dir <assets_directory>
```

5. イメージの準備ができたら、ターゲットマシンを起動し、インストールが完了するまで待ちます。

6. インストールを監視するには、次のコマンドを実行します。

```
$ openshift-install agent wait-for install-complete --dir <assets_directory>
```

注記

完全に機能するハブクラスターを設定するには、次のマニフェストを作成し、コマンド **$ oc apply -f <manifest-name>** を実行して手動で適用する必要があります。マニフェストの作成順序は重要であり、必要に応じて待機状態が表示されます。

7. AI サービスに必要な PV については、次のマニフェストを作成します。

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
 name: assisted-service
 namespace: openshift-local-storage
spec:
 logLevel: Normal
 managementState: Managed
 storageClassDevices:
   - devicePaths:
       - /dev/vda
       - /dev/vdb
     storageClassName: assisted-service
     volumeMode: Filesystem
```

8. 後続のマニフェストを適用する前に、次のコマンドを使用して PV が使用可能になるまで待機します。

```
$ oc wait localvolume -n openshift-local-storage assisted-service --for condition=Available --timeout 10m
```

注記

The `devicePath` is an example and may vary depending on the actual hardware configuration used.

9. マルチクラスターエンジンインスタンスのマニフェストを作成します。

Example MultiClusterEngine.yaml

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec: {}
```

10. マニフェストを作成して AI サービスを有効にします。

例 agentserviceconfig.yaml

```
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
  name: agent
  namespace: assisted-installer
spec:
 databaseStorage:
  storageClassName: assisted-service
  accessModes:
  - ReadWriteOnce
  resources:
   requests:
    storage: 10Gi
 filesystemStorage:
  storageClassName: assisted-service
  accessModes:
  - ReadWriteOnce
  resources:
   requests:
    storage: 10Gi
```

11. 後続のスポーククラスターをデプロイするためのマニフェストを作成します。

例 clusterimageset.yaml

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
```

```
  metadata:
    name: "4.16"
  spec:
    releaseImage: quay.io/openshift-release-dev/ocp-release:4.16.0-x86_64
```

12. マニフェストを作成して、エージェントがインストールされたクラスター (マルチクラスターエンジンと Assisted Service をホストするクラスター) をハブクラスターとしてインポートします。

例 autoimport.yaml

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
 labels:
   local-cluster: "true"
   cloud: auto-detect
   vendor: auto-detect
 name: local-cluster
spec:
 hubAcceptsClient: true
```

13. マネージドクラスターが作成されるまで待ちます。

```
$ oc wait -n multicluster-engine managedclusters local-cluster --for
condition=ManagedClusterJoined=True --timeout 10m
```

検証

- マネージドクラスターのインストールが成功したことを確認するには、次のコマンドを実行します。

```
$ oc get managedcluster
NAME            HUB ACCEPTED   MANAGED CLUSTER URLS            JOINED   AVAILABLE
AGE
local-cluster   true           https://<your cluster url>:6443   True     True       77m
```

関連情報

- ローカルストレージオペレーター

13.6. AGENT-BASED INSTALLER のインストール設定パラメーター

Agent-based Installer を使用して OpenShift Container Platform クラスターをデプロイする前に、クラスターとそれをホストするプラットフォームをカスタマイズするパラメーターを指定します。**install-config.yaml** ファイルと **agent-config.yaml** ファイルの作成時に、必須パラメーターの値を指定する必要があります。オプションのパラメーターを使用してクラスターをさらにカスタマイズできます。

13.6.1. 使用可能なインストール設定パラメーター

次の表に、エージェントベースのインストールプロセスの一部として設定できる必須およびオプションのインストール設定パラメーターを示します。

これらの値は、**install-config.yaml** ファイルに指定されます。

注記

これらの設定はインストールのみに使用され、インストール後は変更できません。

13.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表13.7 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。**install-config.yaml** ファイルまたは **Agent-config.yaml** ファイルのいずれかを介して **metadata.name** を指定しない場合 (たとえば、ZTP マニフェストのみを使用する場合)、クラスター名は **agent-cluster** に設定されます。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform:	インストールの実行に使用する特定プラットフォームの設定: **baremetal**、**external**、**none**、または **vsphere**。	Object
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

13.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

- Red Hat OpenShift Networking OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 と IPv6 の両方のアドレスファミリーがサポートされます。

両方の IP アドレスファミリーを使用するようにクラスターを設定する場合は、次の要件を確認してください。

- どちらの IP ファミリーも、デフォルトゲートウェイに同じネットワークインターフェイスを使用する必要があります。
- 両方の IP ファミリーにデフォルトゲートウェイが必要です。
- すべてのネットワーク設定パラメーターに対して、IPv4 アドレスと IPv6 アドレスを同じ順序で指定する必要があります。たとえば、以下の設定では、IPv4 アドレスは IPv6 アドレスの前に記載されます。

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd00:10:128::/56
    hostPrefix: 64
```

```
serviceNetwork:
- 172.30.0.0/16
- fd00:172:16::/112
```

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表13.8 ネットワークパラメーター

パラメーター	説明	値
`networking:`	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
`networking:` `  networkType:`	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
`networking:` `  clusterNetwork:`	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 `networking:` `  clusterNetwork:` `  - cidr: 10.128.0.0/14` `  hostPrefix: 23` `  - cidr: fd01::/48` `  hostPrefix: 64`
`networking:` `  clusterNetwork:` `  cidr:`	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 OVN-Kubernetes ネットワークプラグインを使用する場合は、IPv4 および IPv6 ネットワークを指定できます。	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。IPv6 ブロックの接頭辞長は **0** から **128** までです。例: **10.128.0.0/14** または **fd01::/48**

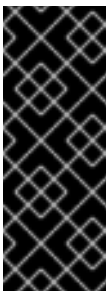
パラメーター	説明	値
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 IPv4 ネットワークの場合、デフォルト値は **23** です。IPv6 ネットワークの場合、デフォルト値は **64** です。デフォルト値は、IPv6 の最小値でもあります。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。 OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 および IPv6 アドレスファミリーの両方に IP アドレスブロックを指定できます。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16 - fd02::/112
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** または **fd00::/48** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

13.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表13.9 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は、**amd64**、**arm64**、**ppc64le**、および **s390x** です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**baremetal**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は、**amd64**、**arm64**、**ppc64le**、および **s390x** です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**

パラメーター	説明	値
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**baremetal**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]

パラメーター	説明	値
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。 **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

13.6.1.4. Agent-based Installer 用の追加のベアメタル設定パラメーター

Agent-based Installer 用の追加のベアメタルインストール設定パラメーターについて、次の表で説明します。

注記

以下のフィールドは、クラスターの初期プロビジョニング中には使用されませんが、クラスターのインストールが完了すると使用できるようになります。インストール時にこれらのフィールドを設定しておくと、Day 2 オペレーション中にフィールドを設定する必要がなくなります。

表13.10 追加のベアメタルパラメーター

パラメーター	説明	値
platform: baremetal: clusterProvisioningIP:	プロビジョニングサービスが実行されるクラスター内の IP アドレス。デフォルトは、プロビジョニングサブネットの 3 番目の IP アドレスに設定されます。たとえば、**172.22.0.3** または **2620:52:0:1307::3 です**。	IPV4 または IPV6 アドレス。
platform: baremetal: provisioningNetwork:	**provisioningNetwork** 設定は、クラスターがプロビジョニングネットワークを使用するかどうかを決定します。存在する場合、設定はクラスターがネットワークを管理するかどうかも決定します。 **Managed**: デフォルト。DHCP、TFTP などのプロビジョニングネットワークを完全に管理するには、このパラメーターを **Managed** に設定します。 **Disabled**: プロビジョニングネットワークの要件を無効にするには、このパラメーターを **Disabled** に設定します。**Disabled** に設定した場合、Day 2 に使用できるプロビジョニングが、仮想メディアベースのプロビジョニングだけになります。**Disabled**にして、電源管理を使用する場合、BMC はベアメタルネットワークからアクセスできる必要があります。Disabled の場合は、プロビジョニングサービスに使用するベアメタルネットワークに 2 つの IP アドレスを指定する必要があります。	**Managed** または **Disabled**。

パラメーター	説明	値
platform: baremetal: provisioningMACAddress:	プロビジョニングサービスを実行するクラスター内の MAC アドレス。	MAC アドレス
platform: baremetal: provisioningNetworkCIDR:	プロビジョニングに使用するネットワークの CIDR。このオプションは、プロビジョニングネットワークでデフォルトのアドレス範囲を使用しない場合に必要です。	有効な CIDR (例: **10.0.0.0/16**)。
platform: baremetal: provisioningNetworkInterface:	ベアメタルネットワークに接続されたノード上のネットワークインターフェイス 名。**provisioningNetworkInterface** 設定を使用して NIC の名前を識別する代わりに、**bootMACAddress** 設定を使用して、Ironic が NIC の IP アドレスを識別できるようにします。	文字列。
platform: baremetal: provisioningDHCPRange:	プロビジョニングネットワーク上のノードの IP 範囲を定義します (例: **172.22.0.10,172.22.0.254**)。	IP アドレスの範囲。
platform: baremetal: hosts:	ベアメタルホストの設定。	ホスト設定オブジェクトの配列。
platform: baremetal: hosts: name:	ホストの名前。	文字列。

パラメーター	説明	値
platform: baremetal: hosts: bootMACAddress:	ホストのプロビジョニングに使用する NIC の MAC アドレス。	MAC アドレス
platform: baremetal: hosts: bmc:	ホストがベースボード管理コントローラー (BMC) に接続するための設定。	BMC 設定オブジェクトのディクショナリー。
platform: baremetal: hosts: bmc: username:	BMC のユーザー名。	文字列。
platform: baremetal: hosts: bmc: password:	BMC のパスワード。	文字列。
platform: baremetal: hosts: bmc: address:	ホストの BMC コントローラーと通信するための URL。address 設定ではプロトコルを指定します。たとえば、**redfish+http://10.10.10.1:8000/redfish/v1/Systems/1234** を指定すると、Redfish が有効になります。詳細は、「インストーラーでプロビジョニングされるクラスターのベアメタルへのデプロイ」セクションの「BMC アドレス指定」を参照してください。	URL。
platform: baremetal: hosts: bmc: disableCertificateVerification:	**redfish** および **redfish-virtualmedia** では、このパラメーターで BMC アドレスを管理する必要があります。BMC アドレスに自己署名証明書を使用する場合は、値が **True** である必要があります。	ブール値。

13.6.1.5. 追加の VMware vSphere 設定パラメーター

追加の VMware vSphere 設定パラメーターは以下の表で説明されています。

表13.11 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform: vsphere:	クラスターをホストするクラウドプラットフォーム上のアカウントについて説明します。パラメーターを使用してプラットフォームをカスタマイズできます。マシンプール内のコンピュートマシンとコントロールプレーンマシンに追加の設定を指定する場合、このパラメーターは必要ありません。OpenShift Container Platform クラスターに指定できる vCenter サーバーは 1 つだけです。	vSphere 設定オブジェクトのディクショナリー
platform: vsphere: failureDomains:	リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。	障害ドメイン設定オブジェクトの配列。
platform: vsphere: failureDomains: name:	障害ドメインの名前。	String
platform: vsphere: failureDomains: region:	クラスターに複数の障害ドメインを定義する場合は、タグを各 vCenter データセンターにアタッチする必要があります。リージョンを定義するには、**openshift-region** タグカテゴリーのタグを使用します。単一の vSphere データセンター環境の場合、タグをアタッチする必要はありませんが、パラメーターに英数字の値 (例: **datacenter**) を入力する必要があります。	String
platform: vsphere: failureDomains: server:	クライアントが障害ドメインリソースにアクセスできるように、VMware vCenter Server の完全修飾ホスト名または IP アドレスを指定します。**server** ロールを vSphere vCenter サーバーの場所に適用する必要があります。	String
platform: vsphere: failureDomains: zone:	クラスターに複数の障害ドメインを定義する場合は、各 vCenter クラスターにタグをアタッチする必要があります。ゾーンを定義するには、**openshift-zone** タグカテゴリーのタグを使用します。単一の vSphere データセンター環境の場合、タグをアタッチする必要はありませんが、パラメーターに英数字の値 (例: **cluster**) を入力する必要があります。	String

パラメーター	説明	値
platform: vsphere: failureDomains: topology: computeCluster:	vSphere コンピュートクラスターへのパス。	String
platform: vsphere: failureDomains: topology: datacenter:	OpenShift Container Platform 仮想マシン (VM) が動作するデータセンターをリストして定義します。データセンターのリストは、**vcenters** フィールドで指定したデータセンターのリストと一致する必要があります。	String
platform: vsphere: failureDomains: topology: datastore:	仮想マシンファイル、テンプレート、ISO イメージを保持する vSphere データストアへのパス。  **重要** データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion はデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage vMotion を無効にする必要があります。 複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。	String
platform: vsphere: failureDomains: topology: folder:	オプション: ユーザーが仮想マシンを作成する既存のフォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。	String

パラメーター	説明	値
platform: vsphere: failureDomains: topology: networks:	設定した仮想 IP アドレスと DNS レコードを含む vCenter インスタンス内のネットワークをリスト表示します。	String
platform: vsphere: failureDomains: topology: resourcePool:	オプション: このパラメーターは、インストールプログラムが仮想マシンを作成する既存のリソースプールの絶対パスを設定します (例: **/<datacenter_name>/host/<cluster_name>/Resources/<resource_pool_name>/<optional_nested_resource_pool_name>**)。	String
platform: vsphere: failureDomains: topology template:	既存の Red Hat Enterprise Linux CoreOS (RHCOS) イメージテンプレートまたは仮想マシンへの絶対パスを指定します。その後、インストールプログラムはイメージテンプレートまたは仮想マシンを使用して、vSphere ホストに RHCOS を迅速にインストールできます。RHCOS イメージを vSphere ホストにアップロードする代わりに、このパラメーターを使用することを検討してください。このパラメーターは、installer-provisioned infrastructure でのみ使用できます。	String
platform: vsphere: vcenters:	サービスが vCenter サーバーと通信できるように接続の詳細を設定します。現在、単一の vCenter サーバーのみサポートされます。	vCenter 設定オブジェクトの配列。
platform: vsphere: vcenters: datacenters:	OpenShift Container Platform 仮想マシン (VM) が動作するデータセンターをリストして定義します。データセンターのリストは、**failureDomains** フィールドで指定されたデータセンターのリストと一致する必要があります。	String
platform: vsphere: vcenters: password:	vSphere ユーザーに関連付けられたパスワード。	String

パラメーター	説明	値
platform: vsphere: vcenters: port:	vCenter サーバーとの通信に使用するポート番号。	Integer
platform: vsphere: vcenters: server:	vCenter サーバーの完全修飾ホスト名 (FQHN) または IP アドレス。	String
platform: vsphere: vcenters: user:	vSphere ユーザーに関連付けられたユーザー名。	String

13.6.1.6. 非推奨の VMware vSphere 設定パラメーター

OpenShift Container Platform 4.13 では、次の vSphere 設定パラメーターが非推奨になりました。これらのパラメーターは引き続き使用できますが、インストールプログラムはこれらのパラメーターを **install-config.yaml** ファイルに自動的に指定しません。

次の表に、非推奨になった各 vSphere 設定パラメーターを示します。

表13.12 非推奨の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform: vsphere: cluster:	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform: vsphere: datacenter:	OpenShift Container Platform 仮想マシン (VM) が動作するデータセンターを定義します。	文字列
platform: vsphere: defaultDatastore:	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列

パラメーター	説明	値
platform: vsphere: folder:	オプション: インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられたフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform: vsphere: password:	vCenter ユーザー名のパスワード。	文字列
platform: vsphere: resourcePool:	オプション: インストールプログラムが仮想マシンを作成する既存のリソースプールの絶対パス。値を指定しない場合、インストールプログラムは **/<datacenter_name>/host/<cluster_name>/Resources** の下のクラスターのルートにリソースをインストールします。	文字列 (例: **/<datacenter_name>/host/<cluster_name>/Resources/<resource_pool_name>/<optional_nested_resource_pool_name>)**。
platform: vsphere: username:	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform: vsphere: vCenter:	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列

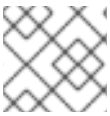
関連情報

- BMC アドレス指定
- VMware vCenter のリージョンとゾーンの設定

13.6.2. 使用可能なエージェント設定パラメーター

次の表に、エージェントベースのインストールプロセスの一部として設定できる必須およびオプションのエージェント設定パラメーターを示します。

これらの値は、**agent-config.yaml** ファイルに指定されています。



注記

これらの設定はインストールのみに使用され、インストール後は変更できません。

13.6.2.1. 必須設定パラメーター

次の表は、必須のエージェント設定パラメーターについて説明しています。

表13.13 必須パラメーター

パラメーター	説明	値
apiVersion:	**agent-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1beta1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。**agent-config.yaml** ファイルに入力された値は無視され、代わりに **install-config.yaml** ファイルに指定された値が使用されます。**install-config.yaml** ファイルまたは **Agent-config.yaml** ファイルのいずれかを介して **metadata.name** を指定しない場合 (たとえば、ZTP マニフェストのみを使用する場合)、クラスター名は **agent-cluster** に設定されます。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。

13.6.2.2. オプションの設定パラメーター

次の表は、オプションのエージェント設定パラメーターについて説明しています。

表13.14 オプションのパラメーター

パラメーター	説明	値

パラメーター	説明	値
rendezvousIP:	ブートストラッププロセスを実行し、**assisted-service** コンポーネントを実行するノードの IP アドレス。**networkConfig** パラメーターで少なくとも 1 つのホストの IP アドレスを指定しない場合は、ランデブー IP アドレスを指定する必要があります。このアドレスが指定されていない場合、指定されたホストの **networkConfig** から 1 つの IP アドレスが選択されます。	IPV4 または IPV6 アドレス。
bootArtifactsBaseURL:	Agent-based Installer を使用して iPXE スクリプトを生成する際に、Preboot Execution Environment (PXE) アセットをアップロードするサーバーの URL。詳細は、「OpenShift Container Platform の PXE アセットの準備」を参照してください。	文字列。
additionalNTPSources:	すべてのクラスターホストに追加される Network Time Protocol (NTP) ソースのリスト。これらは、他の手段で設定された NTP ソースに追加されます。	ホスト名または IP アドレスのリスト。
hosts:	ホストの設定。オプションのホストリスト。定義されたホストの数は、**install-config.yaml** ファイルで定義されたホストの総数 (**compute.replicas** および **controlPlane.replicas** パラメーターの値の合計) を超えてはなりません。	ホスト設定オブジェクトの配列。
hosts: hostname:	ホスト名。動的ホスト設定プロトコル (DHCP) または逆引き DNS ルックアップから取得したホスト名をオーバーライドします。各ホストには、いずれかの手段で指定された一意のホスト名が必要ですが、このパラメーターを使用したホスト名の設定はオプションです。	文字列。
hosts: interfaces:	ホスト上のインターフェイスの名前と MAC アドレスのマッピングテーブルを提供します。**agent-config.yaml** ファイルに **NetworkConfig** セクションが指定されている場合は、このテーブルを含める必要があり、値は **NetworkConfig** セクションで指定されたマッピングと一致する必要があります。	ホスト設定オブジェクトの配列。

パラメーター	説明	値
hosts: interfaces: name:	ホスト上のインターフェイスの名前。	文字列。
hosts: interfaces: macAddress:	ホスト上のインターフェイスの MAC アドレス。	たとえば **00-B0-D0-63-C2-26** などの MAC アドレス。
hosts: role:	ホストが **master** ノードと **worker** ノードのどちらであるかを定義します。**agent-config.yaml** ファイルにロールが定義されていない場合は、クラスターのインストール時にロールがランダムに割り当てられます。	**master** または **worker**。
hosts: rootDeviceHints:	特定デバイスへの Red Hat Enterprise Linux CoreOS (RHCOS) イメージのプロビジョニングを有効にします。インストールプログラムは、検出順にデバイスを検査し、検出された値をヒントの値と比較します。ヒントの値と一致する最初に検出されたデバイスが使用されます。これは、インストール中にオペレーティングシステムが書き込まれるデバイスです。	キーと値のペアのディクショナリー。詳細は、「OpenShift インストール環境のセットアップ」ページの「ルートデバイスのヒント」を参照してください。
hosts: rootDeviceHints: deviceName:	RHCOS イメージがプロビジョニングされるデバイスの名前。	文字列。
hosts: networkConfig: 	ホストネットワーク定義。この設定は、nmstate ドキュメント で定義されている Host Network Management API と一致する必要があります。	ホストネットワーク設定オブジェクトのディクショナリー。

関連情報

- OpenShift Container Platform 用の PXE アセットの準備
- ルートデバイスのヒント

第14章 単一ノードへのインストール

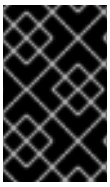
14.1. 単一ノードへのインストールの準備

14.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認している。

14.1.2. 単一ノードでの OpenShift について

標準のインストール方法で単一ノードクラスターを作成できます。単一ノード上の OpenShift Container Platform は、特別な Ignition 設定ファイルの作成を必要とする特殊なインストールです。プライマリーユースケースは、断続的な接続、ポータブルクラウド、および 5G ラジオアクセスネットワーク (RAN) などのエッジコンピューティングのワークロード向けです。単一ノードでのインストールに関する主なトレードオフは、高可用性がないことです。



重要

単一ノードの OpenShift での OpenShiftSDN の使用はサポートされていません。OVN-Kubernetes は、単一ノードの OpenShift デプロイメントのデフォルトのネットワークプラグインです。

14.1.3. 単一ノードに OpenShift をインストールするための要件

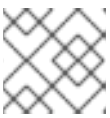
OpenShift Container Platform を単一ノードにインストールすると、高可用性および大規模なクラスターの一部の要件が軽減されます。ただし、以下の要件を満たす必要があります。

- **管理ホスト:** ISO を準備して USB ブートドライブを作成し、インストールを監視するためのコンピューターが必要です。

  注記

  ppc64le プラットフォームの場合、ホストは ISO を準備する必要がありますが、USB ブートドライブを作成する必要はありません。ISO は PowerVM に直接マウントできます。

  注記

  IBM Z® のインストールには ISO は必要ありません。

- **CPU アーキテクチャー:** OpenShift Container Platform をシングルノードにインストールすると、**x86_64**、**arm64**、**ppc64le**、および **s390x** CPU アーキテクチャーがサポートされます。
- **サポートされているプラットフォーム:** シングルノードへの OpenShift Container Platform のインストールは、ベアメタル、vSphere、AWS クラウド、Red Hat OpenStack、OpenShift Virtualization、IBM Power®、および IBM Z® プラットフォームでサポートされています。
- **実稼働環境グレードサーバー:** OpenShift Container Platform を単一ノードにインストールし、OpenShift Container Platform サービスと実稼働のワークロードを実行するのに十分なリソースを持つサーバーが必要です。

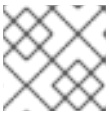
表14.1 最小リソース要件

プロファイル	vCPU	メモリー	ストレージ
最低限	8 vCPU コア	16 GB のメモリー	120 GB

注記

- 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。有効にした場合には、次の式を使用して対応する比率を計算します。
  (コアあたりのスレッド数×コア)×ソケット= vCPU
- インストールプロセス中に Operator を追加すると、最小リソース要件が増加する可能性があります。

仮想メディアを使用して起動する場合は、サーバーには Baseboard Management Controller (BMC) が必要です。



注記

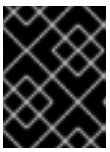
IBM Z® および IBM Power® では、BMC はサポートされていません。

- **ネットワーク**: サーバーは、ルーティング可能なネットワークに接続されていない場合は、インターネットまたはローカルレジストリーにアクセスできるようにする必要があります。サーバーには、Kubernetes API、Ingress ルート、およびクラスターノードドメイン名の DHCP 予約または静的 IP アドレスが必要です。DNS が、以下の完全修飾ドメイン名 (FQDN) のそれぞれに IP アドレスを解決できるように設定する必要があります。

表14.2 必要な DNS レコード

使用法	FQDN	説明
Kubernetes API	**api.<cluster_name>.<base_domain>**	DNS A/AAAA または CNAME レコードを追加します。このレコードは、クラスター外部のクライアントとクラスター内のクライアントの両方で解決可能である必要があります。
Internal API	**api-int.<cluster_name>.<base_domain>**	ISO を手動で作成する場合は、DNS A/AAAA または CNAME レコードを追加します。このレコードは、クラスター内のノードによって解決できる必要があります。

使用法	FQDN	説明
Ingress ルート	***.apps.<cluster_name>.<base_domain>**	ノードをターゲットにするワイルドカード DNS A/AAAA または CNAME レコードを追加します。このレコードは、クラスター外部のクライアントとクラスター内のクライアントの両方で解決可能である必要があります。



重要

永続的な IP アドレスがない場合、**apiserver** と **etcd** の間の通信で失敗する可能性があります。

14.2. 単一ノードでの OPENSHIFT のインストール

Web ベースのアシステッドインストーラーと、アシステッドインストーラーを使用して生成した検出 ISO を使用して、単一ノードの OpenShift をインストールできます。また、**coreos-installer** を使用してインストール ISO を生成することにより、単一ノードの OpenShift をインストールすることもできます。

14.2.1. アシステッドインストーラーを使用した単一ノード OpenShift のインストール

OpenShift Container Platform を単一ノードにインストールするには、Web ベースのアシステッドインストーラーウィザードのガイドに従い、インストールを管理します。

詳細と設定オプションについては、Assisted Installer for OpenShift Container Platform ドキュメントを参照してください。

14.2.1.1. アシステッドインストーラーを使用したディスカバリー ISO の生成

OpenShift Container Platform を単一ノードにインストールするには、アシステッドインストーラーが生成できる検出 ISO が必要です。

手順

1. 管理ホストでブラウザーを開き、Red Hat OpenShift Cluster Manager に移動します。
2. **Create Cluster** をクリックして新規クラスターを作成します。
3. **Cluster Name** フィールドにクラスターの名前を入力します。
4. **Base Domain** フィールドにベースドメインを入力します。以下に例を示します。

   ```
   example.com
   ```

   すべての DNS レコードはこのベースドメインのサブドメインである必要があり、クラスター名が含まれる必要があります。以下に例を示します。

```
<cluster-name>.example.com
```

注記

クラスターのインストール後にベースドメインまたはクラスター名を変更することはできません。

5. **Install single node OpenShift (SNO)** を選択し、ウィザードの残りの手順を完了します。検出 ISO をダウンロードします。
6. 仮想メディアを使用してインストールするための検出 ISO URL を書き留めておきます。

注記

このプロセス中に OpenShift Virtualization を有効にする場合は、仮想マシン用に 50 GiB 以上の 2 つ目のローカルストレージデバイスが必要です。

関連情報

- 論理ボリュームマネージャーストレージを使用した永続ストレージ
- OpenShift Virtualization の機能

14.2.1.2. アシステッドインストーラーを使用した単一ノード OpenShift のインストール

アシステッドインストーラーを使用して、単一ノードクラスターをインストールします。

手順

1. RHCOS 検出 ISO をターゲットホストにアタッチします。
2. サーバーの BIOS 設定で起動ドライブの順序を設定して、アタッチされた検出 ISO から起動し、サーバーを再起動します。
3. 管理ホストで、ブラウザーに戻ります。ホストが、検出されたホストのリストに表示されるまで待ちます。必要に応じて、**Assisted Clusters** ページを再読み込みし、クラスター名を選択します。
4. インストールウィザードの手順を完了します。使用可能なサブネットからのサブネットを含む、ネットワークの詳細を追加します。必要に応じて SSH 公開鍵を追加します。
5. インストールの進捗を監視します。クラスターイベントを確認します。インストールプロセスがサーバーのハードディスクへのオペレーティングシステムイメージの書き込みを完了すると、サーバーが再起動します。
6. 検出 ISO を削除し、インストールドライブから起動するようにサーバーをリセットします。サーバーが自動的に数回再起動し、コントロールプレーンがデプロイされます。

関連情報

- USB ドライブに起動可能な ISO イメージを作成する
- Redfish API を使用した HTTP ホスト ISO イメージからの起動

- 単一ノードの OpenShift クラスターへのワーカーノードの追加

14.2.2. 単一ノードの OpenShift を手動でインストールする

OpenShift Container Platform を単一ノードにインストールするには、最初にインストール ISO を生成してから、ISO からサーバーを起動します。**openshift-install** インストールプログラムを使用して、インストールを監視できます。

関連情報

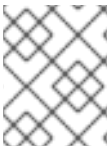
- ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件
- ユーザーによってプロビジョニングされる DNS 要件
- DHCP または静的 IP アドレスの設定

14.2.2.1. coreos-installer によるインストール ISO の生成

OpenShift Container Platform を単一ノードにインストールするには、インストール ISO が必要です。これは、以下の手順で生成できます。

前提条件

- **podman** をインストールします。



注記

DNS レコードを含むネットワーク要件については、「単一ノードに OpenShift をインストールするための要件」を参照してください。

手順

1. OpenShift Container Platform バージョンを設定します。

```
$ OCP_VERSION=<ocp_version> 1
```

1 **<ocp_version>** は、現在のバージョン (**latest-4.16** など) に置き換えます。

2. ホストアーキテクチャーを設定します。

```
$ ARCH=<architecture> 1
```

1 **<architecture>** をターゲットホストアーキテクチャー (**aarch64** や **x86_64** など) に置き換えます。

3. OpenShift Container Platform クライアント (**oc**) をダウンロードし、次のコマンドを入力して使用できるようにします。

```
$ curl -k https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$OCP_VERSION/openshift-client-linux.tar.gz -o oc.tar.gz
```

```
$ tar zxf oc.tar.gz
```

```
$ chmod +x oc
```

4. OpenShift Container Platform インストーラーをダウンロードし、以下のコマンドを入力して使用できるようにします。

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$OCP_VERSION/openshift-install-linux.tar.gz -o openshift-install-linux.tar.gz
```

```
$ tar zxvf openshift-install-linux.tar.gz
```

```
$ chmod +x openshift-install
```

5. 次のコマンドを実行して、RHCOS ISO URL を取得します。

```
$ ISO_URL=$(./openshift-install coreos print-stream-json | grep location | grep $ARCH | grep iso | cut -d\" -f4)
```

6. RHCOS ISO をダウンロードします。

```
$ curl -L $ISO_URL -o rhcos-live.iso
```

7. **install-config.yaml** ファイルを作成します。

```
apiVersion: v1
baseDomain: <domain> (1)
compute:
- name: worker
  replicas: 0 (2)
controlPlane:
  name: master
  replicas: 1 (3)
metadata:
  name: <name> (4)
networking: (5)
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16 (6)
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
bootstrapInPlace:
  installationDisk: /dev/disk/by-id/<disk_id> (7)
pullSecret: '<pull_secret>' (8)
sshKey: |
  <ssh_key> (9)
```

1. クラスタードメイン名を追加します。
2. **compute** レプリカを **0** に設定します。これにより、コントロールプレーンノードがスケジュール可能になります。
3. **controlPlane** レプリカを **1** に設定します。この設定は、以前の **compute** 設定と組み合わせて、クラスターが単一ノードで実行されるようにします。
4. **メタデータ** 名をクラスター名に設定します。
5. **networking** の詳細を設定します。OVN-Kubernetes は、単一ノードクラスターで許可されている唯一のネットワークプラグインタイプです。
6. 単一ノードの OpenShift クラスターのサブネットと一致するように **cidr** 値を設定します。
7. インストールディスクドライブへのパスを設定します (例: **/dev/disk/by-id/wwn-0x64cd98f04fde100024684cf3034da5c2)**。
8. Red Hat OpenShift Cluster Manager からプルシークレット をコピーし、その内容をこの設定に追加します。
9. インストール後にクラスターにログインできるように、管理ホストから公開 SSH 鍵を追加します。

8. 以下のコマンドを実行して、OpenShift Container Platform アセットを生成します。

```
$ mkdir ocp
```

```
$ cp install-config.yaml ocp
```

```
$ ./openshift-install --dir=ocp create single-node-ignition-config
```

9. 以下のコマンドを実行して、Ignition データを RHCOS ISO に埋め込みます。

```
$ alias coreos-installer='podman run --privileged --pull always --rm \
      -v /dev:/dev -v /run/udev:/run/udev -v $PWD:/data \
      -w /data quay.io/coreos/coreos-installer:release'
```

```
$ coreos-installer iso ignition embed -fi ocp/bootstrap-in-place-for-live-iso.ign rhcos-live.iso
```

関連情報

- 単一ノードに OpenShift Container Platform をインストールする方法の詳細は、単一ノードに OpenShift をインストールするための要件 を参照してください。
- インストール前に無効にしたクラスター機能を有効にする方法の詳細は、クラスター機能の有効化 を参照してください。
- 各ケイパビリティーで提供される機能の詳細は、OpenShift Container Platform 4.16 のオプションのクラスター機能 を参照してください。

14.2.2.2. openshift-install を使用したクラスターのインストールの監視

openshift-install を使用して、単一ノードクラスターのインストールの進行状況を監視します。

手順

1. 変更した RHCOS インストール ISO をターゲットホストにアタッチします。
2. サーバーの BIOS 設定で起動ドライブの順序を設定して、アタッチされた検出 ISO から起動し、サーバーを再起動します。
3. 管理ホストで、次のコマンドを実行してインストールを監視します。

   ```
   $ ./openshift-install --dir=ocp wait-for install-complete
   ```

   コントロールプレーンのデプロイ中にサーバーが数回再起動します。

検証

- インストールが完了したら、次のコマンドを実行して環境を確認します。

  ```
  $ export KUBECONFIG=ocp/auth/kubeconfig
  ```

  ```
  $ oc get nodes
  ```

  出力例

  ```
  NAME                         STATUS   ROLES          AGE     VERSION
  control-plane.example.com    Ready    master,worker  10m     v1.29.4
  ```

関連情報

- USB ドライブに起動可能な ISO イメージを作成する
- Redfish API を使用した HTTP ホスト ISO イメージからの起動
- シングルノード OpenShift クラスターへのワーカーノードの追加

14.2.3. クラウドプロバイダーへのシングルノード OpenShift のインストール

14.2.3.1. クラウドプロバイダーにシングルノード OpenShift をインストールするための追加要件

クラウドプロバイダー上でインストーラーによってプロビジョニングされたインストールに関するドキュメントは、3 つのコントロールプレーンノードで設定される高可用性クラスターに基づいています。ドキュメントを参照するときは、単一ノード OpenShift クラスターと高可用性クラスターの要件の違いを考慮してください。

- 高可用性クラスターには、一時的なブートストラップマシン、3 台のコントロールプレーンマシン、および少なくとも 2 台のコンピュートマシンが必要です。シングルノード OpenShift クラスターの場合、必要なのは一時ブートストラップマシンと、コントロールプレーンノード用の 1 つのクラウドインスタンスのみであり、ワーカーノードは必要ありません。

- 高可用性クラスターのインストールの最小リソース要件には、4 つの vCPU と 100 GB のストレージを備えたコントロールプレーンノードが含まれます。単一ノードの OpenShift クラスターの場合、少なくとも 8 つの vCPU コアと 120 GB のストレージが必要です。
- **install-config.yaml** ファイルの **controlPlane.replicas** 設定は **1** に設定する必要があります。
- **install-config.yaml** ファイルの **compute.replicas** 設定は **0** に設定する必要があります。これにより、コントロールプレーンノードがスケジュール可能になります。

14.2.3.2. シングルノード OpenShift でサポートされているクラウドプロバイダー

次の表には、サポートされているクラウドプロバイダーと CPU アーキテクチャーのリストが含まれています。

表14.3 サポートされているクラウドプロバイダー

クラウドプロバイダー	CPU アーキテクチャー
Amazon Web Service (AWS)	x86_64 and AArch64
Microsoft Azure	x86_64
Google Cloud Platform (GCP)	x86_64 and AArch64

14.2.3.3. AWS への単一ノード OpenShift のインストール

単一ノードクラスターを AWS にインストールするには、カスタマイズを使用した AWS へのクラスターのインストール手順を使用して、インストーラーがプロビジョニングしたインストールが必要です。

関連情報

- カスタマイズによる AWS へのクラスターのインストール

14.2.3.4. Azure への単一ノード OpenShift のインストール

単一ノードクラスターを Azure にインストールするには、カスタマイズを使用した Azure へのクラスターのインストール手順を使用して、インストーラーがプロビジョニングしたインストールが必要です。

関連情報

- カスタマイズによる Azure へのクラスターのインストール

14.2.3.5. GCP への単一ノード OpenShift のインストール

単一ノードクラスターを GCP にインストールするには、カスタマイズを使用した GCP へのクラスターのインストール手順を使用して、インストーラーがプロビジョニングしたインストールが必要です。

関連情報

- カスタマイズによる GCP へのクラスターのインストール

14.2.4. USB ドライブに起動可能な ISO イメージを作成する

ISO イメージを含む起動可能な USB ドライブを使用して、ソフトウェアをインストールできます。USB ドライブを使用してサーバーを起動すると、ソフトウェアをインストールするサーバーの準備が整います。

手順

1. 管理ホストで、USB ドライブを USB ポートに挿入します。
2. 起動可能な USB ドライブを作成します。以下に例を示します。

```
# dd if=<path_to_iso> of=<path_to_usb> status=progress
```

ここでは、以下のようになります。

<path_to_iso>
: **rhcos-live.iso** など、ダウンロードした ISO ファイルへの相対パスです。

<path_to_usb>
: **/dev/sdb** など、接続された USB ドライブの場所です。

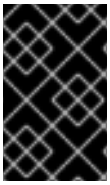
ISO が USB ドライブにコピーされたら、USB ドライブを使用してサーバーにソフトウェアをインストールできます。

14.2.5. Redfish API を使用した HTTP ホスト ISO イメージからの起動

Redfish Baseboard Management Controller (BMC) API を使用してインストールした ISO を使用して、ネットワーク内のホストをプロビジョニングできます。

注記

この手順例では、Dell サーバーでの手順を示します。



重要

ハードウェアと互換性のある iDRAC の最新ファームウェアバージョンがあることを確認してください。ハードウェアまたはファームウェアに問題がある場合は、プロバイダーに連絡する必要があります。

前提条件

- インストール Red Hat Enterprise Linux CoreOS (RHCOS) ISO をダウンロードしている。
- iDRAC9 と互換性のある Dell PowerEdge サーバーを使用している。

手順

1. ネットワークでアクセス可能な HTTP サーバーに ISO ファイルをコピーします。
2. ホストされている ISO ファイルからホストを起動します。以下に例を示します。
   a. 次のコマンドを実行して、Redfish API を呼び出し、ホストされている ISO を **VirtualMedia** ブートメディアとして設定します。

```
$ curl -k -u <bmc_username>:<bmc_password> -d '{"Image":"<hosted_iso_file>",
"Inserted": true}' -H "Content-Type: application/json" -X POST
<host_bmc_address>/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia/CD/Actions/Vi
rtualMedia.InsertMedia
```

ここでは、以下のようになります。

<bmc_username>:<bmc_password>

ターゲットホスト BMC のユーザー名とパスワードです。

<hosted_iso_file>

ホストされたインストール ISO の URL です (例: **http://webserver.example.com/rhcos-live-minimal.iso**)。ISO は、ターゲットホストマシンからアクセスできる必要があります。

<host_bmc_address>

ターゲットホストマシンの BMC IP アドレスです。

b. 次のコマンドを実行して、**VirtualMedia** デバイスから起動するようにホストを設定します。

```
$ curl -k -u <bmc_username>:<bmc_password> -X PATCH -H 'Content-Type:
application/json' -d '{"Boot": {"BootSourceOverrideTarget": "Cd",
"BootSourceOverrideMode": "UEFI", "BootSourceOverrideEnabled": "Once"}}'
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1
```

c. ホストを再起動します。

```
$ curl -k -u <bmc_username>:<bmc_password> -d '{"ResetType": "ForceRestart"}' -H
'Content-type: application/json' -X POST
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.R
eset
```

d. オプション: ホストの電源がオフになっている場合は、**{"ResetType": "On"}** スイッチを使用して起動できます。以下のコマンドを実行します。

```
$ curl -k -u <bmc_username>:<bmc_password> -d '{"ResetType": "On"}' -H 'Content-
type: application/json' -X POST
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.R
eset
```

14.2.6. リモートサーバーアクセス用のカスタムライブ RHCOS ISO の作成

場合によっては、外部ディスクドライブをサーバーに接続することはできませんが、サーバーにリモートアクセスしてノードをプロビジョニングする必要があります。サーバーへの SSH アクセスを有効にすることを推奨します。起動後にサーバーにアクセスできるように、SSHd を有効にし、事前定義された認証情報を使用してライブ RHCOS ISO を作成できます。

前提条件

- **butane** ユーティリティーをインストールしました。

手順

1. **coreos-installer** イメージ ミラー ページから **coreos-installer** バイナリーをダウンロードします。
2. mirror.openshift.com から最新のライブ RHCOS ISO をダウンロードします。
3. **butane** ユーティリティーが Ignition ファイルの作成に使用する **embedded.yaml** ファイルを作成します。

```
variant: openshift
version: 4.16.0
metadata:
  name: sshd
  labels:
    machineconfiguration.openshift.io/role: worker
passwd:
  users:
    - name: core (1)
      ssh_authorized_keys:
        - '<ssh_key>'
```

(1) **core** ユーザーには sudo 権限があります。

4. **butane** ユーティリティーを実行し、以下のコマンドを使用して Ignition ファイルを作成します。

```
$ butane -pr embedded.yaml -o embedded.ign
```

5. Ignition ファイルが作成されたら、**coreos-installer** ユーティリティーを使用して、**rhcos-sshd-4.16.0-x86_64-live.x86_64.iso** という名前の新しいライブ RHCOS ISO に設定を含めることができます。

```
$ coreos-installer iso ignition embed -i embedded.ign rhcos-4.16.0-x86_64-live.x86_64.iso -o rhcos-sshd-4.16.0-x86_64-live.x86_64.iso
```

検証

- 次のコマンドを実行して、カスタムライブ ISO を使用してサーバーを起動できることを確認します。

```
# coreos-installer iso ignition show rhcos-sshd-4.16.0-x86_64-live.x86_64.iso
```

出力例

```
{
  "ignition": {
    "version": "3.2.0"
  },
  "passwd": {
    "users": [
      {
        "name": "core",
        "sshAuthorizedKeys": [
          "ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAADAQABAAABAQCZnG8AIzlDAhpyENpK2qKiTT8EbRWOrz7NXjRzo
pbPu215mocaJgjjwJjh1cYhgPhpAp6M/ttTk7I4OI7g4588Apx4bwJep6oWTU35LkY8ZxkGVPAJL
8kVITdKQviDv3XX12l4QfnDom4tm4gVbRH0gNT1wzhnLP+LKYm2Ohr9D7p9NBnAdro6k++X
WgkDeijLRUTwdEyWunIdW1f8G0Mg8Y1Xzr13BUo3+8aey7HLKJMDtobkz/C8ESYA/f7HJc5Fx
F0XbapWWovSSDJrr9OmlL9f4TfE+cQk3s+eoKiz2bgNPRgEEwihVbGsCN4grA+RzLCAOpec+
2dTJrQvFqsD alosadag@sonnelicht.local"
      ]
    }
  ]
 }
}
```

14.2.7. IBM Z および IBM LinuxONE を使用したシングルノード OpenShift のインストール

IBM Z® および IBM® LinuxONE に単一ノードクラスターをインストールするには、以下の手順のいずれかを使用してユーザーによってプロビジョニングされるインストールが必要です。

- z/VM を使用したクラスターの IBM Z® および IBM® LinuxONE へのインストール
- RHEL KVM を使用したクラスターの IBM Z® および IBM® LinuxONE へのインストール
- IBM Z® および IBM® LinuxONE 上の LPAR へのクラスターのインストール

注記

IBM Z® にシングルノードクラスターをインストールすると、開発環境およびテスト環境のインストールが簡素化され、エントリーレベルで必要なリソース要件が少なくなります。

ハードウェア要件

- クラスターごとに、SMT2 対応の 2 つの Integrated Facilities for Linux (IFL) に相当します。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z® の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。

14.2.7.1. IBM Z および IBM LinuxONE での z/VM を使用したシングルノード OpenShift のインストール

前提条件

- **podman** をインストールしている。

手順

1. 次のコマンドを実行して、OpenShift Container Platform のバージョンを設定します。

```
$ OCP_VERSION=<ocp_version> 1
```

1 **<ocp_version>** を現在のバージョンに置き換えます。たとえば、latest です。

2. 次のコマンドを実行して、ホストアーキテクチャーを設定します。

```
$ ARCH=<architecture> 1
```

1 **<architecture>** をターゲットホストアーキテクチャー **s390x** に置き換えます。

3. OpenShift Container Platform クライアント (**oc**) をダウンロードし、次のコマンドを入力して使用できるようにします。

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/${ARCH}/clients/ocp/${OCP_VERSION}/openshift-client-linux.tar.gz -o oc.tar.gz
```

```
$ tar zxf oc.tar.gz
```

```
$ chmod +x oc
```

4. OpenShift Container Platform インストーラーをダウンロードし、以下のコマンドを入力して使用できるようにします。

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/${ARCH}/clients/ocp/${OCP_VERSION}/openshift-install-linux.tar.gz -o openshift-install-linux.tar.gz
```

```
$ tar zxvf openshift-install-linux.tar.gz
```

```
$ chmod +x openshift-install
```

5. **install-config.yaml** ファイルを作成します。

```
apiVersion: v1
baseDomain: <domain> 1
compute:
- name: worker
  replicas: 0 2
controlPlane:
  name: master
  replicas: 1 3
metadata:
  name: <name> 4
networking: 5
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16 6
  networkType: OVNKubernetes
```

```
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
bootstrapInPlace:
  installationDisk: /dev/disk/by-id/<disk_id> 7
pullSecret: '<pull_secret>' 8
sshKey: |
  <ssh_key> 9
```

1. クラスタードメイン名を追加します。
2. **compute** レプリカを **0** に設定します。これにより、コントロールプレーンノードがスケジュール可能になります。
3. **controlPlane** レプリカを **1** に設定します。この設定は、以前の **compute** 設定と組み合わせて、クラスターが単一ノードで実行されるようにします。
4. **メタデータ** 名をクラスター名に設定します。
5. **networking** の詳細を設定します。OVN-Kubernetes は、単一ノードクラスターで許可されている唯一のネットワークプラグインタイプです。
6. 単一ノードの OpenShift クラスターのサブネットと一致するように **cidr** 値を設定します。
7. インストールディスクドライブへのパスを設定します (例: **/dev/disk/by-id/wwn-0x64cd98f04fde100024684cf3034da5c2)**。
8. Red Hat OpenShift Cluster Manager からプルシークレット をコピーし、その内容をこの設定に追加します。
9. インストール後にクラスターにログインできるように、管理ホストから公開 SSH 鍵を追加します。

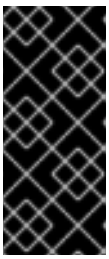
6. 以下のコマンドを実行して、OpenShift Container Platform アセットを生成します。

```
$ mkdir ocp
```

```
$ cp install-config.yaml ocp
```

```
$ ./openshift-install --dir=ocp create single-node-ignition-config
```

7. Red Hat カスタマーポータル の 製品ダウンロード ページまたは RHCOS イメージミラー ページから、RHEL **kernel**、**initramfs**、および **rootfs** アーティファクトを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、**initramfs**、および **rootfs** アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

kernel

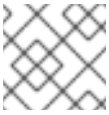
rhcos-<version>-live-kernel-<architecture>

initramfs

rhcos-<version>-live-initramfs.<architecture>.img

rootfs

rhcos-<version>-live-rootfs.<architecture>.img



注記

rootfs イメージは FCP および DASD の場合と同じです。

8. 次のアーティファクトとファイルを HTTP または HTTPS サーバーに移動します。
   - ダウンロードされた RHEL ライブ **kernel**、**initramfs**、および **rootfs** アーティファクト
   - Ignition ファイル
9. 特定の仮想マシンのパラメーターファイルを作成します。

パラメーターファイルの例

```
rd.neednet=1 \
console=ttysclp0 \
coreos.live.rootfs_url=<rhcos_liveos>:8080/rootfs.img \ 1
ignition.firstboot ignition.platform.id=metal \
ignition.config.url=<rhcos_ign>:8080/ignition/bootstrap-in-place-for-live-iso.ign \ 2
ip=encbdd0:dhcp::02:00:00:02:34:02 3
rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
rd.dasd=0.0.4411 \ 4
rd.zfcp=0.0.8001,0x50050763040051e3,0x4000406300000000 \ 5
zfcp.allow_lun_scan=0 \
rd.luks.options=discard
```

1. **coreos.live.rootfs_url=** アーティファクトには、起動している **kernel` および `initramfs** に一致する **rootfs** アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
2. **ignition.config.url=** パラメーターには、マシンロールの Ignition ファイルを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
3. **ip=** パラメーターの場合は、DHCP を使用して自動的に IP アドレスを割り当てるか、「IBM Z® および IBM® LinuxONE での z/VM を使用したクラスターのインストールの説明」に従って手動で IP アドレスを割り当てます。
4. DASD タイプのディスクにインストールする場合は、**rd.dasd=** を使用して、RHCOS がインストールされる DASD を指定します。FCP タイプのディスクの場合は、このエントリーを省略します。
5. FCP タイプのディスクにインストールする場合は、**rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して、RHCOS がインストールされる FCP ディスクを指定します。DASD タイプの

使用して、RHCOS がインストールされる FCP ディスクを指定します。DASD タイプのディスクの場合は、このエントリーを省略します。

その他のパラメーターはすべて変更しません。

10. 次のアーティファクト、ファイル、イメージを z/VM に転送します。たとえば、FTP を使用する場合は以下のようになります。
    - **kernel** と **initramfs** アーティファクト
    - パラメーターファイル
    - RHCOS イメージ
      FTP でファイルを転送し、仮想リーダーから起動する方法については、Z/VM 環境へのインストール を参照してください。
11. ブートストラップノードになる z/VM ゲスト仮想マシンの仮想リーダーに対してファイルの punch を実行します。
12. ブートストラップマシンで CMS にログインします。
13. 次のコマンドを実行して、リーダーからブートストラップマシンを IPL します。

    ```
    $ cp ipl c
    ```

14. 仮想マシンを最初に再起動した後、次のコマンドを直接続けて実行します。
    a. 最初の再起動後に DASD デバイスを起動するには、次のコマンドを実行します。

    ```
    $ cp i <devno> clear loadparm prompt
    ```

    ここでは、以下のようになります。

    <devno>
    ゲストから見えるブートデバイスのデバイス番号を指定します。

    ```
    $ cp vi vmsg 0 <kernel_parameters>
    ```

    ここでは、以下のようになります。

    <kernel_parameters>
    システム制御プログラムデータ (SCPDATA) として保存されるカーネルパラメーターのセットを指定します。Linux をブートするとき、これらのカーネルパラメーターは、ブート設定で使用される既存のカーネルパラメーターの末尾に連結されます。組み合わせたパラメーター文字列は 896 文字を超えてはなりません。

    b. 最初の再起動後に FCP デバイスを起動するには、次のコマンドを実行します。

    ```
    $ cp set loaddev portname <wwpn> lun <lun>
    ```

    ここでは、以下のようになります。

    <wwpn>
    ターゲットポートと **<lun>** 論理ユニットを 16 進数形式で指定します。

```
$ cp set loaddev bootprog <n>
```

ここでは、以下のようになります。

<n>
ブートするカーネルを指定します。

```
$ cp set loaddev scpdata {APPEND|NEW} '<kernel_parameters>'
```

ここでは、以下のようになります。

<kernel_parameters>
システム制御プログラムデータ (SCPDATA) として保存されるカーネルパラメーターのセットを指定します。Linux をブートするとき、これらのカーネルパラメーターは、ブート設定で使用される既存のカーネルパラメーターの末尾に連結されます。組み合わせたパラメーター文字列は 896 文字を超えてはなりません。

<APPEND|NEW>
オプション: カーネルパラメーターを既存の SCPDATA に追加するには、**APPEND** を指定します。これはデフォルトになります。**NEW** を指定して既存の SCPDATA を置き換えます。

例

```
$ cp set loaddev scpdata
'rd.zfcp=0.0.8001,0x500507630a0350a4,0x4000409D00000000
ip=encbdd0:dhcp::02:00:00:02:34:02 rd.neednet=1'
```

IPL およびブートプロセスを開始するには、次のコマンドを実行します。

```
$ cp i <devno>
```

ここでは、以下のようになります。

<devno>
ゲストから見えるブートデバイスのデバイス番号を指定します。

14.2.7.2. IBM Z および IBM LinuxONE での RHEL KVM を使用したシングルノード OpenShift のインストール

前提条件

- **podman** をインストールしている。

手順

1. 次のコマンドを実行して、OpenShift Container Platform のバージョンを設定します。

```
$ OCP_VERSION=<ocp_version> 1
```

1 **<ocp_version>** を現在のバージョンに置き換えます。たとえば、latest です。

2. 次のコマンドを実行して、ホストアーキテクチャーを設定します。

```
$ ARCH=<architecture> 1
```

1 **<architecture>** をターゲットホストアーキテクチャー **s390x** に置き換えます。

3. OpenShift Container Platform クライアント (**oc**) をダウンロードし、次のコマンドを入力して使用できるようにします。

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/${ARCH}/clients/ocp/${OCP_VERSION}/openshift-client-linux.tar.gz -o oc.tar.gz
```

```
$ tar zxf oc.tar.gz
```

```
$ chmod +x oc
```

4. OpenShift Container Platform インストーラーをダウンロードし、以下のコマンドを入力して使用できるようにします。

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/${ARCH}/clients/ocp/${OCP_VERSION}/openshift-install-linux.tar.gz -o openshift-install-linux.tar.gz
```

```
$ tar zxvf openshift-install-linux.tar.gz
```

```
$ chmod +x openshift-install
```

5. **install-config.yaml** ファイルを作成します。

```
apiVersion: v1
baseDomain: <domain> 1
compute:
- name: worker
  replicas: 0 2
controlPlane:
  name: master
  replicas: 1 3
metadata:
  name: <name> 4
networking: 5
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16 6
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
bootstrapInPlace:
```

```
  installationDisk: /dev/disk/by-id/<disk_id> 7
pullSecret: '<pull_secret>' 8
sshKey: |
  <ssh_key> 9
```

1. クラスタードメイン名を追加します。
2. **compute** レプリカを **0** に設定します。これにより、コントロールプレーンノードがスケジュール可能になります。
3. **controlPlane** レプリカを **1** に設定します。この設定は、以前の **compute** 設定と組み合わせて、クラスターが単一ノードで実行されるようにします。
4. ***メタデータ*** 名をクラスター名に設定します。
5. **networking** の詳細を設定します。OVN-Kubernetes は、単一ノードクラスターで許可されている唯一のネットワークプラグインタイプです。
6. 単一ノードの OpenShift クラスターのサブネットと一致するように **cidr** 値を設定します。
7. インストールディスクドライブへのパスを設定します (例: **/dev/disk/by-id/wwn-0x64cd98f04fde100024684cf3034da5c2)**。
8. Red Hat OpenShift Cluster Manager からプルシークレット をコピーし、その内容をこの設定に追加します。
9. インストール後にクラスターにログインできるように、管理ホストから公開 SSH 鍵を追加します。

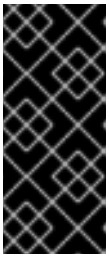
6. 以下のコマンドを実行して、OpenShift Container Platform アセットを生成します。

```
$ mkdir ocp
```

```
$ cp install-config.yaml ocp
```

```
$ ./openshift-install --dir=ocp create single-node-ignition-config
```

7. Red Hat カスタマーポータル の 製品ダウンロード ページまたは RHCOS イメージミラー ページから、RHEL **kernel**、**initramfs**、および **rootfs** アーティファクトを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、**initramfs**、および **rootfs** アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

kernel

rhcos-<version>-live-kernel-<architecture>

initramfs

rhcos-<version>-live-initramfs.<architecture>.img

rootfs

rhcos-<version>-live-rootfs.<architecture>.img

8. **virt-install** を起動する前に、次のファイルとアーティファクトを HTTP または HTTPS サーバーに移動します。
   - ダウンロードされた RHEL ライブ **kernel**、**initramfs**、および **rootfs** アーティファクト
   - Ignition ファイル
9. 次のコンポーネントを使用して、KVM ゲストノードを作成します。
   - RHEL の **kernel** および **initramfs** アーティファクト
   - Ignition ファイル
   - 新しいディスクイメージ
   - 調整された parm ライン引数

```
$ virt-install \
  --name <vm_name> \
  --autostart \
  --memory=<memory_mb> \
  --cpu host \
  --vcpus <vcpus> \
  --location <media_location>,kernel=<rhcos_kernel>,initrd=<rhcos_initrd> \ 1
  --disk size=100 \
  --network network=<virt_network_parm> \
  --graphics none \
  --noautoconsole \
  --extra-args "ip=<ip>::<gateway>:<mask>:<hostname>::none" \
  --extra-args "nameserver=<name_server>" \
  --extra-args "ip=dhcp rd.neednet=1 ignition.platform.id=metal ignition.firstboot" \
  --extra-args "coreos.live.rootfs_url=<rhcos_liveos>" \ 2
  --extra-args "ignition.config.url=<rhcos_ign>" \ 3
  --extra-args "random.trust_cpu=on rd.luks.options=discard" \
  --extra-args "console=ttysclp0" \
  --wait
```

1. **--location** パラメーターには、HTTP または HTTPS サーバー上の kernel/initrd の場所を指定します。
2. **coreos.live.rootfs_url=** アーティファクトには、起動している **kernel** と **initramfs** に一致する **rootfs** アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
3. **ignition.config.url=** パラメーターには、マシンロールの Ignition ファイルを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

14.2.7.3. IBM Z および IBM LinuxONE の LPAR への単一ノード OpenShift のインストール

前提条件

- 単一ノードクラスターをデプロイする場合は、コンピュートノードがゼロの場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

手順

1. 次のコマンドを実行して、OpenShift Container Platform のバージョンを設定します。

```
$ OCP_VERSION=<ocp_version> 1
```

1 **<ocp_version>** を現在のバージョンに置き換えます。たとえば、latest です。

2. 次のコマンドを実行して、ホストアーキテクチャーを設定します。

```
$ ARCH=<architecture> 1
```

1 **<architecture>** をターゲットホストアーキテクチャー **s390x** に置き換えます。

3. OpenShift Container Platform クライアント (**oc**) をダウンロードし、次のコマンドを入力して使用できるようにします。

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/${ARCH}/clients/ocp/${OCP_VERSION}/openshift-client-linux.tar.gz -o oc.tar.gz
```

```
$ tar zxvf oc.tar.gz
```

```
$ chmod +x oc
```

4. OpenShift Container Platform インストーラーをダウンロードし、以下のコマンドを入力して使用できるようにします。

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/${ARCH}/clients/ocp/${OCP_VERSION}/openshift-install-linux.tar.gz -o openshift-install-linux.tar.gz
```

```
$ tar zxvf openshift-install-linux.tar.gz
```

```
$ chmod +x openshift-install
```

5. **install-config.yaml** ファイルを作成します。

```
apiVersion: v1
baseDomain: <domain> 1
```

```
compute:
- name: worker
  replicas: 0 2
controlPlane:
  name: master
  replicas: 1 3
metadata:
  name: <name> 4
networking: 5
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16 6
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
pullSecret: '<pull_secret>' 7
sshKey: |
  <ssh_key> 8
```

1. クラスタードメイン名を追加します。
2. **compute** レプリカを **0** に設定します。これにより、コントロールプレーンノードがスケジュール可能になります。
3. **controlPlane** レプリカを **1** に設定します。この設定は、以前の **compute** 設定と組み合わせて、クラスターが単一ノードで実行されるようにします。
4. **メタデータ** 名をクラスター名に設定します。
5. **networking** の詳細を設定します。OVN-Kubernetes は、単一ノードクラスターで許可されている唯一のネットワークプラグインタイプです。
6. 単一ノードの OpenShift クラスターのサブネットと一致するように **cidr** 値を設定します。
7. Red Hat OpenShift Cluster Manager からプルシークレット をコピーし、その内容をこの設定に追加します。
8. インストール後にクラスターにログインできるように、管理ホストから公開 SSH 鍵を追加します。

6. 以下のコマンドを実行して、OpenShift Container Platform アセットを生成します。

```
$ mkdir ocp
```

```
$ cp install-config.yaml ocp
```

7. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

8. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. 以下の **spec** スタンザに示されるように **mastersSchedulable** パラメーターを見つけ、これが **true** に設定されていることを確認します。

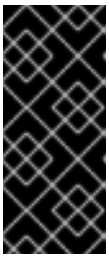
```
spec:
  mastersSchedulable: true
status: {}
```

   c. ファイルを保存し、終了します。
9. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

10. Red Hat カスタマーポータル の 製品ダウンロード ページまたは RHCOS イメージミラー ページから、RHEL **kernel**、**initramfs**、および **rootfs** アーティファクトを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、**initramfs**、および **rootfs** アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

kernel

rhcos-<version>-live-kernel-<architecture>

initramfs

rhcos-<version>-live-initramfs.<architecture>.img

rootfs

rhcos-<version>-live-rootfs.<architecture>.img

注記

rootfs イメージは FCP および DASD の場合と同じです。

11. 次のアーティファクトとファイルを HTTP または HTTPS サーバーに移動します。
    - ダウンロードされた RHEL ライブ **kernel**、**initramfs**、および **rootfs** アーティファクト
    - Ignition ファイル
12. LPAR にブートストラップのパラメーターファイルを作成します。

ブートストラップマシンのパラメーターファイルの例

```
cio_ignore=all,!condev rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/<block_device> \1
coreos.inst.ignition_url=http://<http_server>/bootstrap.ign \2
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \3
ip=<ip>::<gateway>:<netmask>:<hostname>::none nameserver=<dns> \4
rd.znet=qeth,0.0.1140,0.0.1141,0.0.1142,layer2=1,portno=0 \
rd.dasd=0.0.4411 \5
rd.zfcp=0.0.8001,0x50050763040051e3,0x4000406300000000 \6
zfcp.allow_lun_scan=0
```

1. インストール先のシステムのブロックデバイスを指定します。DASD タイプのディスクへのインストールには **dasda** を使用します。FCP タイプのディスクへのインストールには、**sda** を使用します。
2. **bootstrap.ign** 設定ファイルの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
3. **coreos.live.rootfs_url=** アーティファクトには、起動している **kernel` および `initramfs** に一致する **rootfs** アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
4. **ip=** パラメーターの場合、「IBM Z® および IBM® LinuxONE への LPAR へのインストール」の説明に従って、IP アドレスを手動で割り当てます。
5. DASD タイプのディスクにインストールする場合は、**rd.dasd=** を使用して、RHCOS がインストールされる DASD を指定します。FCP タイプのディスクの場合は、このエントリーを省略します。
6. FCP タイプのディスクにインストールする場合は、**rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して、RHCOS がインストールされる FCP ディスクを指定します。DASD タイプのディスクの場合は、このエントリーを省略します。

必要に応じて、追加のパラメーターを調整できます。

13. LPAR にコントロールプレーンのパラメーターファイルを作成します。

コントロールプレーンマシンのパラメーターファイルの例

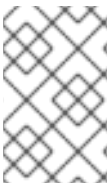
```
cio_ignore=all,!condev rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/<block_device> \
coreos.inst.ignition_url=http://<http_server>/master.ign \ 1
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \
ip=<ip>::<gateway>:<netmask>:<hostname>::none nameserver=<dns> \
rd.znet=qeth,0.0.1140,0.0.1141,0.0.1142,layer2=1,portno=0 \
rd.dasd=0.0.4411 \
rd.zfcp=0.0.8001,0x50050763040051e3,0x4000406300000000 \
zfcp.allow_lun_scan=0
```

1 **master.ign** 設定ファイルの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

14. 次のアーティファクト、ファイル、イメージを z/VM に転送します。たとえば、FTP を使用する場合は以下のようになります。
    - **kernel** と **initramfs** アーティファクト
    - パラメーターファイル
    - RHCOS イメージ
      FTP でファイルを転送して起動する方法については、LPAR へのインストール を参照してください。
15. ブートストラップマシンを起動します。
16. コントロールプレーンマシンを起動します。

14.2.8. IBM Power を使用した単一ノード OpenShift のインストール

IBM Power® に単一ノードクラスターをインストールするには、IBM Power® を使用したクラスターのインストール手順を使用して、ユーザーがプロビジョニングしたインストールが必要です。

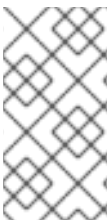


注記

IBM Power® にシングルノードクラスターをインストールすると、開発環境およびテスト環境のインストールが簡素化され、エントリーレベルで必要なリソース要件が少なくなります。

ハードウェア要件

- クラスターごとに、SMT2 対応の 2 つの Integrated Facilities for Linux (IFL) に相当します。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。



注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Power® の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。

関連情報

- クラスターの IBM Power® へのインストール

14.2.8.1. IBM Power を使用した単一ノード OpenShift の基礎のセットアップ

IBM Power® に単一ノード OpenShift をインストールする前に、踏み台をセットアップする必要があります。IBM Power® 上でシングルノード OpenShift の踏み台サーバーをセットアップするには、以下のサービスの設定が必要です。

- PXE は、単一ノード OpenShift クラスターのインストールに使用されます。PXE では、次のサービスを設定して実行する必要があります。
  - API、api-int、および *.apps を定義する DNS
  - PXE を有効にし、単一ノードの OpenShift ノードに IP アドレスを割り当てる DHCP サービス
  - Ignition および RHCOS rootfs イメージを提供する HTTP
  - TFTP による PXE の有効化
- DNS、DHCP、PXE、HTTP の httpd をサポートするには、**dnsmasq** をインストールする必要があります。

これらの要件を満たす踏み台サーバーを設定するには、次の手順を使用します。

手順

1. 次のコマンドを使用して **grub2** をインストールします。これは PowerVM の PXE を有効にするために必要です。

```
grub2-mknetdir --net-directory=/var/lib/tftpboot
```

/var/lib/tftpboot/boot/grub2/grub.cfg ファイルの例

```
default=0
fallback=1
timeout=1
if [ ${net_default_mac} == fa:b0:45:27:43:20 ]; then
menuentry "CoreOS (BIOS)" {
  echo "Loading kernel"
  linux "/rhcos/kernel" ip=dhcp rd.neednet=1 ignition.platform.id=metal ignition.firstboot
coreos.live.rootfs_url=http://192.168.10.5:8000/install/rootfs.img
ignition.config.url=http://192.168.10.5:8000/ignition/sno.ign
  echo "Loading initrd"
  initrd  "/rhcos/initramfs.img"
}
fi
```

2. 次のコマンドを使用して、PXE のミラーリポジトリーから RHCOS イメージファイルをダウンロードします。
   a. 次のコマンドを入力して、**RHCOS_URL** 変数に次の 4.12 URL を割り当てます。

```
$ export RHCOS_URL=https://mirror.openshift.com/pub/openshift-
v4/ppc64le/dependencies/rhcos/4.12/latest/
```

b. 次のコマンドを入力して、**/var/lib/tftpboot/rhcos** ディレクトリーに移動します。

```
$ cd /var/lib/tftpboot/rhcos
```

c. 次のコマンドを実行して、**RHCOS_URL** 変数に保存されている URL から指定された RHCOS カーネルファイルをダウンロードします。

```
$ wget ${RHCOS_URL}/rhcos-live-kernel-ppc64le -o kernel
```

d. 次のコマンドを入力して、**RHCOS_URL** 変数に保存されている URL から RHCOS **initramfs** ファイルをダウンロードします。

```
$ wget ${RHCOS_URL}/rhcos-live-initramfs.ppc64le.img -o initramfs.img
```

e. 次のコマンドを実行して、**/var//var/www/html/install/** ディレクトリーに移動します。

```
$ cd /var//var/www/html/install/
```

f. 次のコマンドを入力して、**RHCOS_URL** 変数に保存されている URL から RHCOS **root filesystem** イメージファイルをダウンロードして保存します。

```
$ wget ${RHCOS_URL}/rhcos-live-rootfs.ppc64le.img -o rootfs.img
```

3. 単一ノード OpenShift クラスターの ignition ファイルを作成するには、**install-config.yaml** ファイルを作成する必要があります。

a. 次のコマンドを入力して、ファイルを保持する作業ディレクトリーを作成します。

```
$ mkdir -p ~/sno-work
```

b. 次のコマンドを入力して、~/**sno-work** ディレクトリーに移動します。

```
$ cd ~/sno-work
```

c. 次のサンプルファイルを使用して、必要な **install-config.yaml** を ~/**sno-work** ディレクトリーに作成できます。

```
apiVersion: v1
baseDomain: <domain> 1
compute:
- name: worker
  replicas: 0 2
controlPlane:
  name: master
  replicas: 1 3
metadata:
  name: <name> 4
networking: 5
  clusterNetwork:
```

```
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16 6
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
bootstrapInPlace:
  installationDisk: /dev/disk/by-id/<disk_id> 7
pullSecret: '<pull_secret>' 8
sshKey: |
  <ssh_key> 9
```

1. クラスタードメイン名を追加します。
2. **compute** レプリカを **0** に設定します。これにより、コントロールプレーンノードがスケジュール可能になります。
3. **controlPlane** レプリカを **1** に設定します。この設定は、以前の **compute** 設定と組み合わせて、クラスターが単一ノードで実行されるようにします。
4. ***メタデータ*** 名をクラスター名に設定します。
5. **networking** の詳細を設定します。OVN-Kubernetes は、単一ノードクラスターで許可されている唯一のネットワークプラグインタイプです。
6. 単一ノードの OpenShift クラスターのサブネットと一致するように **cidr** 値を設定します。
7. インストールディスクドライブへのパスを設定します (例: **/dev/disk/by-id/wwn-0x64cd98f04fde100024684cf3034da5c2)**。
8. Red Hat OpenShift Cluster Manager からプルシークレット をコピーし、その内容をこの設定に追加します。
9. インストール後にクラスターにログインできるように、管理ホストから公開 SSH 鍵を追加します。

4. **openshift-install** イメージをダウンロードして ignition ファイルを作成し、それを **http** ディレクトリーにコピーします。
   a. 次のコマンドを実行して、**openshift-install-linux-4.12.0** .tar ファイルをダウンロードします。

```
$ wget https://mirror.openshift.com/pub/openshift-v4/ppc64le/clients/ocp/4.12.0/openshift-install-linux-4.12.0.tar.gz
```

   b. 次のコマンドを入力して、**openshift-install-linux-4.12.0.tar.gz** アーカイブを展開します。

```
$ tar xzvf openshift-install-linux-4.12.0.tar.gz
```

   c. 以下のコマンドを入力し、以下を実行します。

```
$ ./openshift-install --dir=~/sno-work create create single-node-ignition-config
```

d. 次のコマンドを入力して、Ignition ファイルを作成します。

```
$ cp ~/sno-work/single-node-ignition-config.ign /var/www/html/ignition/sno.ign
```

e. 次のコマンドを入力して、**/var/www/html** ディレクトリーの SELinux ファイルを復元します。

```
$ restorecon -vR /var/www/html || true
```

踏み台には必要なファイルがすべて含まれており、単一ノードの OpenShift をインストールするために適切に設定されています。

14.2.8.2. IBM Power を使用した単一ノード OpenShift のインストール

前提条件

- 踏み台を設定しました。

手順

単一ノード OpenShift クラスターのインストールには 2 つの手順があります。まず、単一ノードの OpenShift 論理パーティション (LPAR) を PXE で起動する必要があり、次にインストールの進行状況を監視する必要があります。

1. 次のコマンドを使用して、netboot で powerVM を起動します。

```
$ lpar_netboot -i -D -f -t ent -m <sno_mac> -s auto -d auto -S <server_ip> -C <sno_ip> -G
<gateway> <lpar_name> default_profile <cec_name>
```

ここでは、以下のようになります。

sno_mac
シングルノード OpenShift クラスターの MAC アドレスを指定します。

sno_ip
単一ノード OpenShift クラスターの IP アドレスを指定します。

server_ip
踏み台 (PXE サーバー) の IP アドレスを指定します。

gateway
ネットワークのゲートウェイ IP を指定します。

lpar_name
HMC 内の単一ノード OpenShift lpar 名を指定します。

cec_name
sno_lpar が存在するシステム名を指定します。

2. 単一ノード OpenShift LPAR が PXE で起動した後、**openshift-install** コマンドを使用してインストールの進行状況を監視します。

a. ブートストラップが完了したら、次のコマンドを実行します。

```
./openshift-install wait-for bootstrap-complete
```

b. 正常に戻ったら、次のコマンドを実行します。

```
./openshift-install wait-for install-complete
```

第15章 ベアメタルへのインストール

15.1. ベアメタルクラスターのインストールの準備

15.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認している。

15.1.2. OpenShift 仮想化のためのベアメタルクラスターの計画

OpenShift 仮想化を使用する場合は、ベアメタルクラスターをインストールする前に、いくつかの要件を認識することが重要です。

- ライブマイグレーション機能を使用する場合は、**クラスターのインストール時に** 複数のワーカーノードが必要です。これは、ライブマイグレーションではクラスターレベルの高可用性 (HA) フラグを true に設定する必要があるためです。HA フラグは、クラスターのインストール時に設定され、後で変更することはできません。クラスターのインストール時に定義されたワーカーノードが 2 つ未満の場合、クラスターの存続期間中、HA フラグは false に設定されます。

  注記

  単一ノードのクラスターに OpenShift Virtualization をインストールできますが、単一ノードの OpenShift は高可用性をサポートしていません。

- ライブマイグレーションには共有ストレージが必要です。OpenShift Virtualization のストレージは、ReadWriteMany (RWX) アクセスモードをサポートし、使用する必要があります。
- Single Root I/O Virtualization (SR-IOV) を使用する予定の場合は、ネットワークインターフェイスコントローラー (NIC) が OpenShift Container Platform でサポートされていることを確認してください。

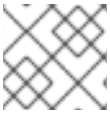
関連情報

- OpenShift Virtualization の開始
- OpenShift Virtualization のクラスターの準備
- Single Root I/O Virtualization (SR-IOV) ハードウェアネットワークについて
- 仮想マシンの SR-IOV ネットワークへの接続

15.1.3. SR-IOV デバイスの NIC パーティショニング (テクノロジープレビュー)

OpenShift Container Platform は、デュアルポートのネットワークインターフェイスカード (NIC) を搭載したサーバーにデプロイできます。1 つの高速デュアルポート NIC を複数の仮想機能 (VF) に分割し、SR-IOV を有効にすることができます。

この機能は、Link Aggregation Control Protocol (LACP) による高可用性のための結合の使用をサポートします。



注記

物理 NIC で宣言できる LACP は 1 つだけです。

OpenShift Container Platform クラスターは、以下の方法を使用して、2 つの物理機能 (PF) に 2 つの VF を持つ結合インターフェイスにデプロイできます。

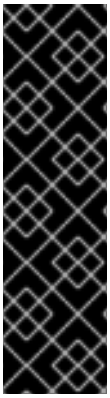
- Agent-based Installer

注記

nmstate の最低限必要なバージョンは次のとおりです。

  - RHEL 8 バージョンの **1.4.2-4**
  - RHEL 9 バージョンの **2.2.7**

- installer-provisioned infrastructure によるインストール
- user-provisioned infrastructure によるインストール



重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

関連情報

- 例: ボンディングと SR-IOV デュアル NIC ノードのネットワーク設定
- オプション: デュアルポート NIC 用のホストネットワークインターフェイスの設定
- 複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合する

15.1.4. ベアメタルに OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform インストールプログラムは、クラスターをデプロイするための 4 つの方法を提供します。

- **インタラクティブ**: Web ベースの Assisted Installer を使用してクラスターをデプロイできます。これは、ネットワークがインターネットに接続されているクラスターに推奨されるアプローチです。Assisted Installer は、OpenShift Container Platform をインストールする最も簡単な方法であり、スマートなデフォルトを提供し、クラスターをインストールする前に事前検証を実行します。また、自動化および高度な設定シナリオのための RESTful API も提供します。
- **ローカルエージェントベース**: エアギャップネットワークまたはネットワークが制限された環境用の Agent-based Installer を使用して、クラスターをローカルにデプロイできます。この方法

では、Assisted Installer の多くの利点を得られますが、最初に Agent-based Installer をダウンロードして設定する必要があります。設定はコマンドラインインターフェイスで行います。このアプローチは、エアギャップまたは制限されたネットワークに最適です。

- **自動化**: インストーラーによってプロビジョニングされたインフラストラクチャーとそれが維持するクラスターにクラスターをデプロイできます。インストーラーは、各クラスターホストのベースボード管理コントローラー (BMC) をプロビジョニングに使用します。接続されたネットワーク、エアギャップされたネットワーク、または制限されたネットワークの両方でクラスターをデプロイできます。

- **完全な制御**: お客様が準備および保守するインフラストラクチャーにクラスターをデプロイメントできます。これにより、最大限のカスタマイズ性が提供されます。接続されたネットワーク、エアギャップされたネットワーク、または制限されたネットワークの両方でクラスターをデプロイできます。

クラスターには次の特徴があります。

- 単一障害点のない高可用性インフラストラクチャーがデフォルトで利用可能である。

- 管理者は適用される更新内容および更新タイミングを制御できる。

インストーラーによるプロビジョニングおよびユーザーによるプロビジョニングのインストールプロセスの詳細は、インストールプロセス を参照してください。

15.1.4.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法を使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされるベアメタルインフラストラクチャーに、クラスターをインストールできます。

- **インストーラーでプロビジョニングされるクラスターのベアメタルへのインストール**: インストーラーのプロビジョニングを使用して、OpenShift Container Platform をベアメタルにインストールできます。

15.1.4.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、独自にプロビジョニングするベアメタルインフラストラクチャーに、クラスターをインストールできます。

- **ユーザーによってプロビジョニングされるクラスターのベアメタルへのインストール**: OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールできます。ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

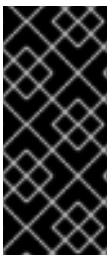
- **ネットワークのカスタマイズを使用したユーザーによってプロビジョニングされるベアメタルクラスターのインストール**: ネットワークのカスタマイズを使用して、ユーザーによってプロビジョニングされるインフラストラクチャーにベアメタルクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。ネットワークのカスタマイズのほとんどは、インストールステージで適用する必要があります。

- **ネットワークが制限された環境でのユーザーによってプロビジョニングされるベアメタルクラスターのインストール**: ミラーレジストリーを使用して、ユーザーによってプロビジョニングされるベアメタルクラスターを制限されたネットワークまたは非接続ネットワークにインストー

ルできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

15.2. ユーザーによってプロビジョニングされるクラスターのベアメタルへのインストール

OpenShift Container Platform 4.16 では、独自にプロビジョニングするベアメタルインフラストラクチャーにクラスターをインストールできます。



重要

以下の手順に従って仮想化環境またはクラウド環境にクラスターをデプロイすることができますが、ベアメタルプラットフォーム以外の場合は追加の考慮事項に注意してください。このような環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

15.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

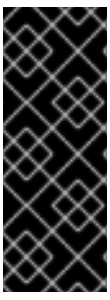
プロキシーを設定する場合は、このサイトリストも確認してください。

15.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

関連情報

- ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーへのネットワークが制限された環境でのインストールの実行についての詳細は、ネットワークが制限された環境でのユーザーによってプロビジョニングされるベアメタルクラスターのインストール を参照してください。

15.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

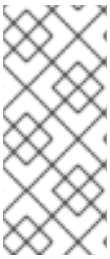
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

15.2.3.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表15.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



注記

例外として、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターで実行できます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。1 つのコンピュートマシンの実行はサポートされていません。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティング

マシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

15.2.3.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表15.2 最小リソース要件

マシン	オペレーティングシステム	CPU [1]	RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. CPU 1 つ分は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = CPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

15.2.3.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- ベアメタル環境での 3 ノードクラスターのデプロイに関する詳細は、3 ノードクラスターの設定 を参照してください。
- インストール後のクラスター証明書署名要求の承認についての詳細は、マシンの証明書署名要求の承認 を参照してください。

15.2.3.4. vSphere 上のベアメタルクラスターの要件

クラスター内のすべての仮想マシンで、**disk.EnableUUID** パラメーターを必ず有効にしてください。

関連情報

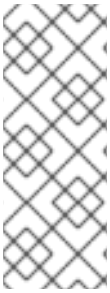
- user-provisioned infrastructure の VMware vSphere 上で **disk.EnableUUID** パラメーターの値を **TRUE** に設定する方法について、詳細は RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始 を参照してください。

15.2.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。



注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

15.2.3.5.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

15.2.3.5.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表15.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP)

サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

15.2.3.6. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

15.2.3.6.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例15.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 (1)
api-int.ocp4.example.com. IN A 192.168.1.5 (2)
;
*.apps.ocp4.example.com. IN A 192.168.1.5 (3)
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 (4)
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 (5)
control-plane1.ocp4.example.com. IN A 192.168.1.98 (6)
control-plane2.ocp4.example.com. IN A 192.168.1.99 (7)
;
compute0.ocp4.example.com. IN A 192.168.1.11 (8)
compute1.ocp4.example.com. IN A 192.168.1.7 (9)
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例15.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

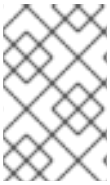
PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

15.2.3.7. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

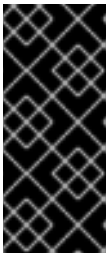


注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

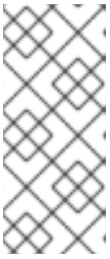
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

15.2.3.7.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例15.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
```

```
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (5)
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (6)
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

(1) ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

(2)(4) ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

15.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。
   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

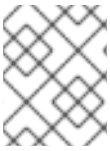
b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件
- RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始
- DHCP を使用したクラスターノードのホスト名の設定
- 高度な RHCOS インストール設定
- ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件
- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

15.2.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   注記

   出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

   random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

   ```
   $ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
   <cluster_name>.<base_domain>
   ```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

関連情報

- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

15.2.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

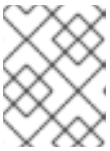
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

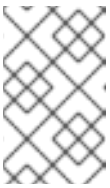
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

関連情報

- ノードの健全性の確認

15.2.7. インストールプログラムの取得

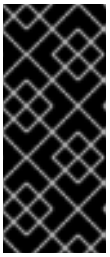
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

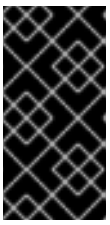
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

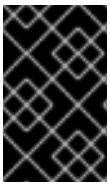
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15.2.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

15.2.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

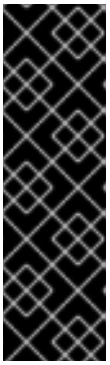
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

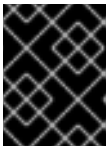
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- ベアメタルのインストール設定パラメーター

15.2.9.1. ベアメタルのサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

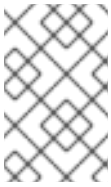
```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
```

```
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OVNKubernetes 11
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths": ...}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



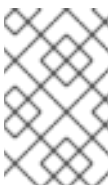
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



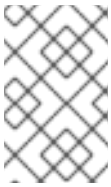
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。プラットフォーム用に追加のプラットフォーム設定変数を指定することはできません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

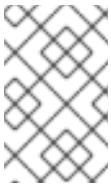
重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

関連情報

- API およびアプリケーションの Ingress 負荷分散要件の詳細は、ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件 を参照してください。
- インストール前に無効にしたクラスター機能を有効にする方法の詳細は、クラスター機能の有効化 を参照してください。
- 各ケイパビリティーで提供される機能の詳細は、OpenShift Container Platform 4.16 のオプションのクラスター機能 を参照してください。

15.2.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

注記

ベアメタルインストールでは、**install-config.yaml** ファイルの **networking.machineNetwork[].cidr** フィールドで指定される範囲にあるノード IP アドレスを割り当てない場合、それらを **proxy.noProxy** フィールドに含める必要があります。

前提条件

- 既存の **install-config.yaml** ファイルがある。

- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

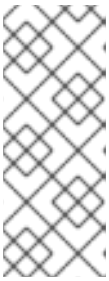
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-**

bundle 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



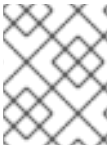
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

15.2.9.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定されるベアメタルクラスターに、ゼロコンピュートマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

15.2.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

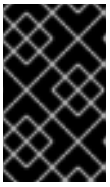
1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

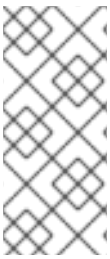
関連情報

- kubelet 証明書のリカバリーに関する詳細は、コントロールプレーン証明書の期限切れの状態からのリカバリー を参照してください。

15.2.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

RHCOS をマシンにインストールするには、ISO イメージまたはネットワーク PXE ブートを使用する手順のいずれかを実行します。



注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL8 コンピュートマシンのみがサポートされています。

以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブインストール起動プロセスを中断してカーネル引数を追加できます。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。

- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールするノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。こ

の特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。

注記

OpenShift Container Platform 4.6 の時点で、RHCOS ISO およびその他のインストールアーティファクトは、4K セクターのディスクへのインストールをサポートします。

15.2.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

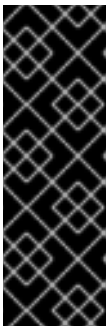
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。

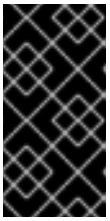
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後、システムを再起動する必要があります。システムの再起動後、指定した Ignition 設定ファイルを適用します。
10. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

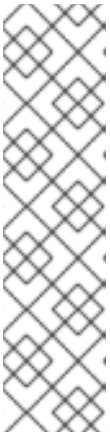
```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

11. 継続してクラスターの他のマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

15.2.11.2. PXE または iPXE ブートを使用した RHCOS のインストール

PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE または iPXE インフラストラクチャーを設定していること。

- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

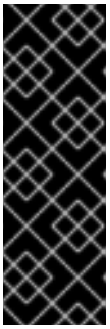
3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
```

```
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs**、**kernel**、および **initramfs** ファイルを HTTP サーバーにアップロードします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。
6. RHCOS イメージの PXE または iPXE インストールを設定し、インストールを開始します。ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

- PXE(**x86_64**) の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
```

```
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

- iPXE (**x86_64** + **aarch64**) の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

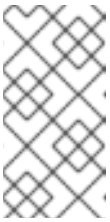
(1) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。



注記

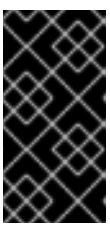
aarch64 アーキテクチャーで CoreOS **kernel** をネットワークブートするには、**IMAGE_GZIP** オプションが有効になっているバージョンの iPXE ビルドを使用する必要があります。iPXE の **IMAGE_GZIP** オプション を参照してください。

- **aarch64** 上の PXE (第 2 段階として UEFI と Grub を使用) の場合:

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
    initrd rhcos-<version>-live-initramfs.<architecture>.img 3
}
```

1. HTTP/TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は、TFTP サーバー上の **kernel** ファイルの場所になります。**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値は HTTP サーバー上のブートストラップ Ignition 設定ファイルの場所になります。
2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。
3. TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

7. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

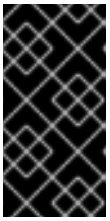
8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

9. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

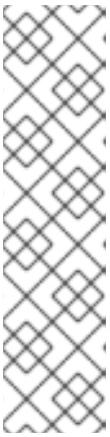
10. クラスターのマシンの作成を続行します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

15.2.11.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。本セクションでは、以下のような手法で実行できるいくつかの設定について説明します。

- カーネル引数をライブインストーラーに渡す
- ライブシステムからの **coreos-installer** の手動による実行
- ライブ ISO または PXE ブートイメージのカスタマイズ

本セクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

15.2.11.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

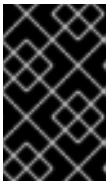
- 詳細の RHCOS インストールリファレンスの表を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

手順

1. ISO インストーラーを起動します。
2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。
3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```



重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

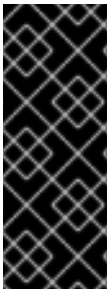
4. インストール済みのシステムで再起動します。

関連情報

- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

15.2.11.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。デフォルトのパーティション設定をオーバーライドしない限り、特定のアーキテクチャーの各 RHCOS ノードで同じパーティションレイアウトが使用されます。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。



重要

ノードでカスタムパーティションスキームを使用すると、OpenShift Container Platform が一部のノードパーティションでモニタリングやアラートを行わなくなる可能性があります。デフォルトのパーティション設定をオーバーライドする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

OpenShift Container Platform は、次の 2 つのファイルシステム識別子を監視します。

- **nodefs**: **/var/lib/kubelet** を含むファイルシステム
- **imagefs**: **/var/lib/containers** を含むファイルシステム

デフォルトのパーティションスキームの場合、**nodefs** と **imagefs** は同じルートファイルシステム (/) を監視します。

RHCOS を OpenShift Container Platform クラスターノードにインストールするときにデフォルトのパーティション設定をオーバーライドするには、別のパーティションを作成する必要があります。コンテナーとコンテナーイメージ用に別のストレージパーティションを追加する状況を考えてみましょう。たとえば、**/var/lib/containers** を別のパーティションにマウントすると、kubelet が **/var/lib/containers** を **imagefs** ディレクトリーとして、ルートファイルシステムを **nodefs** ディレクトリーとして個別に監視します。

重要

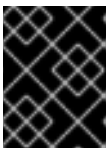
より大きなファイルシステムをホストするためにディスクサイズを変更した場合は、別の **/var/lib/containers** パーティションを作成することを検討してください。多数の割り当てグループによって発生する CPU 時間の問題を軽減するには、**xfs** 形式のディスクのサイズを変更することを検討してください。

15.2.11.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。



重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大

を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

1. インストールホストで、OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

15.2.11.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、 **coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。

注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/disk/by-id/scsi-<serial_number>
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/disk/by-id/scsi-<serial_number>
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/disk/by-id/scsi-<serial_number>
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

15.2.11.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。

重要

これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **Live install Ignition config**: このタイプは、**coreos-installer customize** サブコマンドとそのさまざまなオプションを使用して作成できます。この方法では、Ignition 設定はライブインストールメディアに渡され、起動直後に実行され、RHCOS システムがディスクにインストールされる前または後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。
  PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、 Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

15.2.11.3.4. デフォルトのコンソール設定

OpenShift Container Platform 4.16 ブートイメージからインストールされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードは、ほとんどの仮想化セットアップおよびベアメタルセットアップに対応するためのデフォルトコンソールを使用します。クラウドおよび仮想化プラットフォームが異なれば、選択したアーキテクチャーに応じて、異なるデフォルト設定が使用される場合があります。ベアメタルインストールではカーネルのデフォルト設定が使用されます。これは通常、グラフィカルコンソールがプライマリーコンソールで、シリアルコンソールが無効になっていることを意味します。

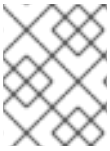
デフォルトのコンソールが特定のハードウェア設定と一致しない場合や、デフォルトのコンソールを調整する必要がある特定のニーズがある場合があります。以下に例を示します。

- デバッグ目的で、コンソールの緊急シェルにアクセスしたいと考えています。
- クラウドプラットフォームは、グラフィカルコンソールへの対話型アクセスを提供しませんが、シリアルコンソールを提供します。
- 複数のコンソールを有効にしたい。

コンソール設定は、ブートイメージから継承されます。これは、既存のクラスター内の新しいノードが、デフォルトのコンソールへの変更の影響を受けないことを意味します。

次の方法で、ベアメタルインストール用にコンソールを設定できます。

- コマンドラインで手動で **coreos-installer** を使用する。
- **--dest-console** オプションを指定して **coreos-installer iso Customize** または **coreos-installer pxe Customize** サブコマンドを使用して、プロセスを自動化するカスタムイメージを作成します。



注記

高度なカスタマイズを行うには、カーネル引数ではなく、**coreos-installer iso** または **coreos-installer pxe** サブコマンドを使用してコンソール設定を実行します。

15.2.11.3.5. PXE および ISO インストール用のシリアルコンソールの有効化

デフォルトでは、Red Hat Enterprise Linux CoreOS (RHCOS) シリアルコンソールは無効になっており、すべての出力はグラフィカルコンソールに書き込まれます。ISO インストール用にシリアルコンソールを有効にし、シリアルコンソールとグラフィカルコンソールの両方に出力が送信されるようにブートローダーを再設定できます。

手順

1. ISO インストーラーを起動します。
2. **coreos-installer** コマンドを実行してシステムをインストールし、**--console** オプションを 1 回追加してグラフィカルコンソールを指定し、2 回目にシリアルコンソールを指定します。

```
$ coreos-installer install \
  --console=tty0 \ 1
  --console=ttyS0,<options> \ 2
  --ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```

1. 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。
2. 望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **11520n8** です。オプションが指定されていない場合、デフォルトのカーネル値である **9600n8** が使用されます。このオプションの形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。

3. インストール済みのシステムで再起動します。

注記

coreos-installer install --append-karg オプションを使用し、**console=** でコンソールを指定すると、同様の結果が得られます。ただし、これはカーネルのコンソールのみを設定し、ブートローダーは設定しません。

PXE インストールを設定するには、**coreos.inst.install_dev** カーネルコマンドラインオプションが省略されていることを確認し、シェルプロンプトを使用して、上記の ISO インストール手順を使用して手動で **coreos-installer** を実行します。

15.2.11.3.6. ライブ RHCOS ISO または PXE インストールのカスタマイズ

ライブ ISO イメージまたは PXE 環境を使用して、Ignition 設定ファイルをイメージに直接挿入することで RHCOS をインストールできます。これにより、システムのプロビジョニングに使用できるカスタマイズされたイメージが作成されます。

ISO イメージの場合、これを行うメカニズムは **coreos-installer iso customize** サブコマンドです。これは設定に合わせて **.iso** ファイルを変更します。同様に、PXE 環境のメカニズムは、カスタマイズを含む新しい **initramfs** ファイルを作成する **coreos-installer pxe customize** サブコマンドです。

customize サブコマンドは、他のタイプのカスタマイズも埋め込むことができる汎用ツールです。次のタスクは、より一般的なカスタマイズの例です。

- 企業のセキュリティーポリシーで使う必要がある場合に備えて、カスタム CA 証明書を挿入します。
- カーネル引数を必要とせずにネットワーク設定を設定します。
- 任意のプレインストールおよびポストインストールスクリプトまたはバイナリーを埋め込みます。

15.2.11.3.7. ライブ RHCOS ISO イメージのカスタマイズ

coreos-installer iso customize サブコマンドを使用して、ライブ RHCOS ISO イメージを直接カスタマイズできます。ISO イメージを起動すると、カスタマイズが自動的に適用されます。

この機能を使用して、RHCOS を自動的にインストールするように ISO イメージを設定できます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページと Ignition 設定ファイルから RHCOS ISO イメージを取得し、次のコマンドを実行して、Ignition 設定を ISO イメージに直接挿入します。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
    --dest-ignition bootstrap.ign \ 1
    --dest-device /dev/disk/by-id/scsi-<serial_number> 2
```

1. **openshift-installer** インストールプログラムから生成される Ignition 設定ファイル。
2. このオプションを指定すると、ISO イメージは自動的にインストールを実行します。それ以外の場合は、イメージはインストール用に設定されたままになります が、**coreos.inst.install_dev** カーネル引数を指定しない限り、自動的にはインストールされません。

3. オプション: ISO イメージのカスタマイズを削除し、イメージを元の状態に戻すには、次のコマンドを実行します。

```
$ coreos-installer iso reset rhcos-<version>-live.x86_64.iso
```

これで、ライブ ISO イメージを再カスタマイズしたり、元の状態で使用したりできます。

カスタマイズを適用すると、それ以降のすべての RHCOS 起動に影響します。

15.2.11.3.7.1. ライブインストール ISO イメージを変更して、シリアルコンソールを有効化

OpenShift Container Platform 4.12 以降でインストールされたクラスターでは、シリアルコンソールはデフォルトで無効になり、すべての出力がグラフィカルコンソールに書き込まれます。次の手順でシリアルコンソールを有効にできます。

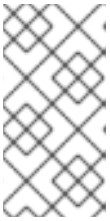
手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。

2. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して ISO イメージをカスタマイズし、シリアルコンソールが出力を受信できるようにします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
  --dest-ignition <path> \ 1
  --dest-console tty0 \ 2
  --dest-console ttyS0,<options> \ 3
  --dest-device /dev/disk/by-id/scsi-<serial_number> 4
```

1. インストールする Ignition 設定の場所。
2. 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。
3. 望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **115200n8** です。オプションが指定されていない場合、デフォルトのカーネル値である **9600n8** が使用されます。このオプションの形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。
4. インストール先として指定されたディスク。このオプションを省略すると、ISO イメージはインストールプログラムを自動的に実行しますが、**coreos.inst.install_dev** カーネル引数も指定しない限り失敗します。



注記

--dest-console オプションは、ライブ ISO システムではなく、インストールされたシステムに影響します。ライブ ISO システムのコンソールを変更するには、**--live-karg-append** オプションを使用し、**console=** でコンソールを指定します。

カスタマイズが適用され、ISO イメージの後続のすべての起動に影響します。

3. オプション: ISO イメージのカスタマイズを削除してイメージを元の状態に戻すには、次のコマンドを実行します。

```
$ coreos-installer iso reset rhcos-<version>-live.x86_64.iso
```

ライブ ISO イメージを再カスタマイズするか、元の状態で使用できるようになりました。

15.2.11.3.7.2. カスタム認証局を使用するようにライブインストール ISO イメージを変更する

customize サブコマンドの **--ignition-ca** フラグを使用して、認証局 (CA) 証明書を Ignition に提供できます。CA 証明書は、インストールの起動時とインストール済みシステムのプロビジョニング時の両方で使用できます。

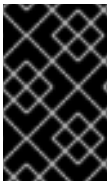
注記

カスタム CA 証明書は、Ignition がリモートリソースをフェッチする方法に影響しますが、システムにインストールされている証明書には影響しません。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して、カスタム CA で使用する ISO イメージをカスタマイズします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso --ignition-ca cert.pem
```



重要

coreos.inst.ignition_url カーネルパラメーターは、**--ignition-ca** フラグでは機能しません。クラスターごとにカスタマイズされたイメージを作成するには、**--dest-ignition** フラグを使用する必要があります。

カスタム CA 証明書を適用すると、それ以降のすべての RHCOS 起動に影響します。

15.2.11.3.7.3. カスタマイズされたネットワーク設定を使用したライブインストール ISO イメージの変更

NetworkManager キーファイルをライブ ISO イメージに埋め込み、**customize** サブコマンドの **--network-keyfile** フラグを使用してインストール済みシステムに渡すことができます。

警告

接続プロファイルを作成する際は、接続プロファイルのファイル名に **.nmconnection** ファイル名拡張子を使用する必要があります。**.nmconnection** ファイル名拡張子を使用しない場合、クラスターは接続プロファイルをライブ環境に適用しますが、クラスターが初めてノードを起動するときに設定が適用されないため、セットアップが機能しなくなります。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. ボンディングされたインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0.nmconnection** ファイルを作成します。

```
[connection]
id=bond0
type=bond
interface-name=bond0
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[bond]
```

```
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto

[proxy]
```

3. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em1.nmconnection** ファイルを作成します。

```
[connection]
id=em1
type=ethernet
interface-name=em1
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

4. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em2.nmconnection** ファイルを作成します。

```
[connection]
id=em2
type=ethernet
interface-name=em2
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

5. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して、設定されたネットワークで ISO イメージをカスタマイズします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
    --network-keyfile bond0.nmconnection \
    --network-keyfile bond0-proxy-em1.nmconnection \
    --network-keyfile bond0-proxy-em2.nmconnection
```

ネットワーク設定はライブシステムに適用され、宛先システムに引き継がれます。

15.2.11.3.8. ライブ RHCOS PXE 環境のカスタマイズ

coreos-installer pxe customize サブコマンドを使用して、ライブ RHCOS PXE 環境を直接カスタマイズできます。PXE 環境を起動すると、カスタマイズが自動的に適用されます。

この機能を使用して、RHCOS を自動的にインストールするように PXE 環境を設定できます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラーページと Ignition 設定ファイルから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、Ignition 設定からのカスタマイズを含む新しい **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
  --dest-ignition bootstrap.ign \ 1
  --dest-device /dev/disk/by-id/scsi-<serial_number> \ 2
  -o rhcos-<version>-custom-initramfs.x86_64.img 3
```

1. **openshift-installer** から生成された Ignition 設定ファイル。
2. このオプションを指定すると、PXE 環境で自動的にインストールが実行されます。それ以外の場合は、イメージはインストール用に設定されたままになりますが、**coreos.inst.install_dev** カーネル引数を指定しない限り、自動的には設定されません。
3. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。

カスタマイズを適用すると、それ以降のすべての RHCOS 起動に影響します。

15.2.11.3.8.1. ライブインストール PXE 環境を変更して、シリアルコンソールを有効化。

OpenShift Container Platform 4.12 以降でインストールされたクラスターでは、シリアルコンソールはデフォルトで無効になり、すべての出力がグラフィカルコンソールに書き込まれます。次の手順でシリアルコンソールを有効にできます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページおよび Ignition 設定ファイルから RHCOS **kernel**、**initramfs** および **rootfs** ファイルを取得します。次のコマンドを実行して、シリアルコンソールが出力を受信できるようにする新しいカスタマイズされた **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
  --dest-ignition <path> \ 1
  --dest-console tty0 \ 2
  --dest-console ttyS0,<options> \ 3
  --dest-device /dev/disk/by-id/scsi-<serial_number> \ 4
  -o rhcos-<version>-custom-initramfs.x86_64.img 5
```

1. インストールする Ignition 設定の場所。
2. 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。
3. 望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **115200n8** です。オプションが指定されていない場合、デフォルトのカーネル値である **9600n8** が使用されます。このオプションの形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。
4. インストール先として指定されたディスク。このオプションを省略すると、PXE 環境は自動的にインストーラーを実行しますが、**coreos.inst.install_dev** カーネル引数も指定しない限り失敗します。
5. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。

カスタマイズが適用され、PXE 環境の後続のすべての起動に影響します。

15.2.11.3.8.2. カスタム認証局を使用するようにライブインストール PXE 環境を変更する

customize サブコマンドの **--ignition-ca** フラグを使用して、認証局 (CA) 証明書を Ignition に提供できます。CA 証明書は、インストールの起動時とインストール済みシステムのプロビジョニング時の両方で使用できます。

注記

カスタム CA 証明書は、Ignition がリモートリソースをフェッチする方法に影響しますが、システムにインストールされている証明書には影響しません。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、カスタム CA で使用するための新しいカスタマイズされた **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
    --ignition-ca cert.pem \
    -o rhcos-<version>-custom-initramfs.x86_64.img
```

3. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。

重要

coreos.inst.ignition_url カーネルパラメーターは、**--ignition-ca** フラグでは機能しません。クラスターごとにカスタマイズされたイメージを作成するには、**--dest-ignition** フラグを使用する必要があります。

カスタム CA 証明書を適用すると、それ以降のすべての RHCOS 起動に影響します。

15.2.11.3.8.3. カスタマイズされたネットワーク設定を使用したライブインストール PXE 環境の変更

NetworkManager キーファイルをライブ PXE 環境に埋め込み、**customize** サブコマンドの **--network-keyfile** フラグを使用して、インストール済みシステムに渡すことができます。

警告

接続プロファイルを作成する際は、接続プロファイルのファイル名に **.nmconnection** ファイル名拡張子を使用する必要があります。**.nmconnection** ファイル名拡張子を使用しない場合、クラスターは接続プロファイルをライブ環境に適用しますが、クラスターが初めてノードを起動するときに設定が適用されないため、セットアップが機能しなくなります。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. ボンディングされたインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0.nmconnection** ファイルを作成します。

```
[connection]
id=bond0
type=bond
interface-name=bond0
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto

[proxy]
```

3. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em1.nmconnection** ファイルを作成します。

```
[connection]
id=em1
type=ethernet
interface-name=em1
```

```
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

4. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em2.nmconnection** ファイルを作成します。

```
[connection]
id=em2
type=ethernet
interface-name=em2
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

5. RHCOS イメージミラー ページから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、設定済みのネットワークを含む新しいカスタマイズされた **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
    --network-keyfile bond0.nmconnection \
    --network-keyfile bond0-proxy-em1.nmconnection \
    --network-keyfile bond0-proxy-em2.nmconnection \
    -o rhcos-<version>-custom-initramfs.x86_64.img
```

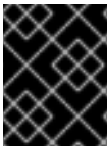
6. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。
ネットワーク設定はライブシステムに適用され、宛先システムに引き継がれます。

15.2.11.3.9. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

15.2.11.3.9.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。



重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

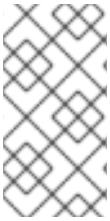
ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**

- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

  ```
  ip=::10.10.10.254::::
  ```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

  ```
  rd.route=20.20.20.0/24:20.20.20.254:enp2s0
  ```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、**options** はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

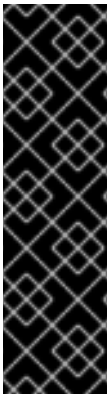
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合する



重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

オプション: **bond=** オプションを使用して、複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合できます。

各ノードで、次のタスクを実行する必要があります。

1. SR-IOV デバイスの管理 のガイダンスに従って、SR-IOV 仮想機能 (VF) を作成します。「仮想マシンへの SR-IOV ネットワークデバイスの接続」セクションの手順に従います。

2. ボンドを作成し、目的の VF をボンドに接続し、ネットワークボンディングの設定 のガイダンスに従って、ボンドリンクの状態を設定します。説明されている手順のいずれかに従って、結合を作成します。

次の例は、使用する必要がある構文を示しています。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は仮想機能 (VF) をカーネル内の既知の名前で表し、**ip link** コマンド (**eno1f0**、**eno2f0**) の出力に表示されます。**options** は結合オプションのコンマ区切りリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1**、**em2**) のコンマ区切りリストを表します。

注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

15.2.11.3.9.2. ISO および PXE インストール用のcoreos-installer オプション

RHCOS は、ISO イメージから RHCOS ライブ環境に起動した後に、コマンドプロンプトで **coreos-installer install <options> <device>** を実行してインストールできます。

以下の表は、**coreos-installer** コマンドに渡すことのできるサブコマンド、オプションおよび引数を示しています。

表15.9 coreos-installer サブコマンド、コマンドラインオプション、および引数

coreos-installer install サブコマンド	
サブコマンド	**説明**
$ coreos-installer install <options> <device>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer install サブコマンドオプション	
オプション	**説明**
-u、--image-url <url>	イメージの URL を手動で指定します。
-f、--image-file <path>	ローカルイメージファイルを手動で指定します。デバッグに使用されます。
-i、--ignition-file <path>	ファイルから Ignition 設定を埋め込みます。
-I、--ignition-url <URL>	URL から Ignition 設定を埋め込みます。
--ignition-hash <digest>	Ignition 設定の **type-value** をダイジェスト値を取得します。
-p、--platform <name>	インストール済みシステムの Ignition プラットフォーム ID を上書きします。
--console <spec>	インストールされたシステムのカーネルとブートローダーコンソールを設定します。**<spec>** の形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。

--append-karg <arg>...	インストール済みシステムにデフォルトのカーネル引数を追加します。
--delete-karg <arg>...	インストール済みシステムからデフォルトのカーネル引数を削除します。
-n、--copy-network	インストール環境からネットワーク設定をコピーします。  **重要** **--copy-network** オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。
--network-dir <path>	**-n** を指定して使用する場合。デフォルトは **/etc/NetworkManager/system-connections/** です。
--save-partlabel <lx>..	このラベル glob でパーティションを保存します。
--save-partindex <id>...	この数または範囲でパーティションを保存します。
--insecure	RHCOS イメージ署名の検証を省略します。
--insecure-ignition	HTTPS またはハッシュなしで Ignition URL を許可します。
--architecture <name>	ターゲット CPU アーキテクチャー。有効な値は **x86_64** および **aarch64** です。
--preserve-on-error	エラー時のパーティションテーブルは消去しないでください。
-h、--help	ヘルプ情報を表示します。
coreos-installer インストールサブコマンド引数	
引数	**説明**
<device>	宛先デバイス。
coreos-installer ISO サブコマンド	
サブコマンド	**説明**

$ coreos-installer iso customize <options> <ISO_image>	RHCOS ライブ ISO イメージをカスタマイズします。
coreos-installer iso reset <options> <ISO_image>	RHCOS ライブ ISO イメージをデフォルト設定に復元します。
coreos-installer iso ignition remove <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を削除します。

coreos-installer ISO カスタマイズサブコマンドオプション

オプション	説明
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。
--dest-karg-append <arg>	宛先システムの各起動にカーネル引数を追加します。
--dest-karg-delete <arg>	宛先システムの各起動からカーネル引数を削除します。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。
--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
--post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
--live-karg-append <arg>	ライブ環境の各ブートにカーネル引数を追加します。

--live-karg-delete <arg>	ライブ環境の各ブートからカーネル引数を削除します。
--live-karg-replace <k=o=n>	ライブ環境の各起動で、**key=old=new** の形式でカーネル引数を置き換えます。
-f, --force	既存の Ignition 設定を上書きします。
-o、--output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。
coreos-installer PXE サブコマンド	
サブコマンド	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
coreos-installer pxe customize <options> <path>	RHCOS ライブ PXE ブート設定をカスタマイズします。
coreos-installer pxe ignition wrap <options>	イメージに Ignition 設定をラップします。
coreos-installer pxe ignition unwrap <options> <image_name>	イメージでラップされた Ignition 設定を表示します。
coreos-installer PXE はサブコマンドオプションをカスタマイズします	
オプション	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。

--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
-o, --output <path>	initramfs を新しい出力ファイルに書き込みます。  **注記** このオプションは、PXE 環境に必要です。
-h、--help	ヘルプ情報を表示します。

15.2.11.3.9.3. ISO または PXE インストールのcoreos.inst ブートオプション

coreos.inst ブートパラメーターを RHCOS ライブインストーラーに渡して、ブート時に **coreos-installer** オプションを自動的に起動できます。これらは、標準のブート引数の追加として提供されます。

- ISO インストールの場合、ブートローダーメニューで自動ブートを中断して **coreos.inst** オプションを追加できます。**RHEL CoreOS (Live)** メニューオプションが強調表示されている状態で **TAB** を押すと、自動ブートを中断できます。
- PXE または iPXE インストールの場合、RHCOS ライブインストーラーのブート前に **coreos.inst** オプションを **APPEND** 行に追加する必要があります。

以下の表は、ISO および PXE インストールの RHCOS ライブインストーラーの **coreos.inst** ブートオプションを示しています。

表15.10 coreos.inst ブートオプション

引数	説明
coreos.inst.install_dev	必須。インストール先のシステムのブロックデバイス。**sda** は許可されていますが、**/dev/sda** などの完全パスを使用することが推奨されます。

引数	説明
coreos.inst.ignition_url	オプション: インストール済みシステムに埋め込む Ignition 設定の URL。URL が指定されていない場合、Ignition 設定は埋め込まれません。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.save_partlabel	オプション: インストール時に保存するパーティションのコンマ区切りのラベル。glob 形式のワイルドカードが許可されます。指定したパーティションは存在する必要はありません。
coreos.inst.save_partindex	オプション: インストール時に保存するパーティションのコンマ区切りのインデックス。範囲 **m-n** は許可され、**m** または **n** のいずれかを省略できます。指定したパーティションは存在する必要はありません。
coreos.inst.insecure	オプション: **coreos.inst.image_url** で署名なしと指定される OS イメージを許可します。
coreos.inst.image_url	オプション: 指定した RHCOS イメージをダウンロードし、インストールします。 ● この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。 ● この引数は、ライブメディアに一致しないバージョンの RHCOS をインストールするために使用できますが、インストールするバージョンに一致するメディアを使用することが推奨されます。 ● **coreos.inst.image_url** を使用している場合は、**coreos.inst.insecure** も使用する必要があります。これは、ベアメタルメディアが OpenShift Container Platform について GPG で署名されていないためです。 ● HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.skip_reboot	オプション: システムはインストール後に再起動しません。インストールが完了するとプロンプトが表示され、インストール時に生じる内容を検査できます。この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。

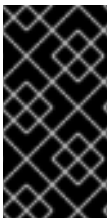
引数	説明
coreos.inst.platform_id	オプション: RHCOS イメージがインストールされるプラットフォームの Ignition プラットフォーム ID。デフォルトは **metal** です。このオプションは、VMware などのクラウドプロバイダーから Ignition 設定を要求するかどうかを決定します。例: **coreos.inst.platform_id=vmware**
ignition.config.url	オプション: ライブ起動の Ignition 設定の URL。たとえば、これは **coreos-installer** の起動方法をカスタマイズしたり、インストール前後にコードを実行するために使用できます。これはインストール済みシステムの Ignition 設定である **coreos.inst.ignition_url** とは異なります。

15.2.11.4. RHCOS のカーネル引数でのマルチパスの有効化

RHCOS はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。

OpenShift Container Platform 4.8 以降でプロビジョニングされたノードのマルチパスを有効にできます。インストール後のサポートは、マシン設定を使用してマルチパスをアクティベートすることで利用できますが、インストール中にマルチパスを有効にすることが推奨されます。

非最適化パスに対して I/O があると、I/O システムエラーが発生するように設定するには、インストール時にマルチパスを有効にする必要があります。



重要

IBM Z® および IBM® LinuxONE では、インストール時にクラスターを設定した場合のみマルチパスを有効にできます。詳細は、**IBM Z® および IBM® LinuxONE への z/VM を使用したクラスターのインストール**の RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始を参照してください。

以下の手順では、インストール時にマルチパスを有効にし、**coreos-installer install** コマンドにカーネル引数を追加して、インストール済みシステム自体が初回起動からマルチパスを使用できるようにします。

注記

OpenShift Container Platform は、4.6 以前からアップグレードされたノードでの day-2 アクティビティーとしてのマルチパスの有効化をサポートしません。

手順

1. マルチパスを有効にして **multipathd** デーモンを起動するには、インストールホストで次のコマンドを実行します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

- 必要に応じて、PXE または ISO を起動する場合は、カーネルコマンドラインから **rd.multipath=default** を追加することで、マルチパスを有効にできます。

2. **coreos-installer** プログラムを呼び出してカーネル引数を追加します。

- マシンに接続されているマルチパスデバイスが 1 つしかない場合は、このデバイスは **/dev/mapper/mpatha** のパスで利用できます。以下に例を示します。

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1 1 つのマルチパスデバイスのパスを指定します。

- 複数のマルチパスデバイスがマシンに接続している場合には、より明示的に **/dev/mapper/mpatha** を使用する代わりに、**/dev/disk/by-id** で利用可能な World Wide Name (WWN) シンボリックリンクを使用することが推奨されます。以下に例を示します。

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1 マルチパス化されたデバイスの WWN ID を指定します。例: **0xx194e957fcedb4841**

特別な **coreos.inst.*** 引数を使用してライブインストーラーを指定する場合に、このシンボリックリンクを **coreos.inst.install_dev** カーネル引数として使用することもできます。詳細は、Installing RHCOS and starting the OpenShift Container Platform bootstrap process を参照してください。

3. ワーカーノードのいずれかに移動し、カーネルコマンドライン引数 (ホストの **/proc/cmdline** 内) をリスト表示してカーネル引数が機能することを確認します。

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

出力例

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

追加したカーネル引数が表示されるはずです。

15.2.11.5. iSCSI ブートデバイスへの RHCOS の手動インストール

RHCOS を iSCSI ターゲットに手動でインストールできます。

前提条件

1. RHCOS ライブ環境を使用している。
2. RHCOS をインストールする iSCSI ターゲットがある。

手順

1. 次のコマンドを実行して、ライブ環境から iSCSI ターゲットをマウントします。

```
$ iscsiadm \
    --mode discovery \
    --type sendtargets
    --portal <IP_address> \ 1
    --login
```

1 ターゲットポータルの IP アドレス

2. 次のコマンドを実行し、必要なカーネル引数を使用して、RHCOS を iSCSI ターゲットにインストールします。以下に例を示します。

```
$ coreos-installer install \
    /dev/disk/by-path/ip-<IP_address>:<port>-iscsi-<target_iqn>-lun-<lun> \ 1
    --append-karg rd.iscsi.initiator=<initiator_iqn> \ 2
    --append.karg netroot=<target_iqn> \ 3
    --console ttyS0,115200n8
    --ignition-file <path_to_file>
```

1 インストール先です。ターゲットポータルの IP アドレス、関連するポート番号、ターゲット iSCSI ノードを IQN 形式、および iSCSI 論理ユニット番号(LUN)で指定する必要があります。

2 IQN 形式の iSCSI イニシエーターまたはクライアントの名前。iSCSI イニシエーターは iSCSI ターゲットに接続するセッションを形成します。

3 IQN 形式の iSCSI ターゲットまたはサーバーの名前。

dracut でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

3. 次のコマンドで、iSCSI ディスクをアンマウントします。

```
$ iscsiadm --mode node --logoutall=all
```

この手順は、**coreos-installer iso customize** または **coreos-installer pxe customize** サブコマンドを使用して実行することもできます。

15.2.11.6. iBFT を使用した iSCSI ブートデバイスへの RHCOS のインストール

完全にディスクレスマシンでは、iSCSI ターゲットとイニシエーターの値を iBFT を介して渡すことができます。iSCSI マルチパスもサポートされています。

前提条件

1. RHCOS ライブ環境を使用している。
2. RHCOS をインストールする iSCSI ターゲットがある。
3. オプション：iSCSI ターゲットをマルチパス化している。

手順

1. 次のコマンドを実行して、ライブ環境から iSCSI ターゲットをマウントします。

```
$ iscsiadm \
    --mode discovery \
    --type sendtargets
    --portal <IP_address> \ 1
    --login
```

1 ターゲットポータルの IP アドレス

2. オプション：以下のコマンドでマルチパスを有効にし、デーモンを起動します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

3. 次のコマンドを実行し、必要なカーネル引数を使用して、RHCOS を iSCSI ターゲットにインストールします。以下に例を示します。

```
$ coreos-installer install \
    /dev/mapper/mpatha \ 1
    --append-karg rd.iscsi.firmware=1 \ 2
    --append-karg rd.multipath=default \ 3
    --console ttyS0 \
    --ignition-file <path_to_file>
```

1 1 つのマルチパスデバイスのパス。複数のマルチパスデバイスが接続されている場合、または明示的な場合は、**/dev/disk/by-path** で利用可能な World Wide Name (WWN)シンボリックリンクを使用できます。

2 iSCSI パラメーターは、BIOS ファームウェアから読み込まれます。

3 オプション：マルチパスを有効にする場合は、このパラメーターを含めます。

dracut でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

4. iSCSI ディスクをアンマウントします。

```
$ iscsiadm --mode node --logout=all
```

この手順は、**coreos-installer iso customize** または **coreos-installer pxe customize** サブコマンドを使用して実行することもできます。

関連情報

- 特別な **coreos.inst.*** 引数を使用してライブインストーラーを指示する方法は、RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始 を参照してください。

15.2.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

関連情報

- インストールログの監視と、インストールの問題が発生した場合の診断データの取得に関する詳細は、インストールの進捗の監視 を参照してください。

15.2.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.2.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

手順

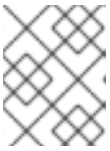
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
master-2 Ready    master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

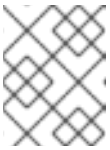
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

15.2.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                            4.16.0    True        False         False      40m
cluster-autoscaler                          4.16.0    True        False         False      37m
config-operator                             4.16.0    True        False         False      38m
```

```
console                                    4.16.0    True        False        False      26m
csi-snapshot-controller                    4.16.0    True        False        False      37m
dns                                        4.16.0    True        False        False      37m
etcd                                       4.16.0    True        False        False      36m
image-registry                             4.16.0    True        False        False      31m
ingress                                    4.16.0    True        False        False      30m
insights                                   4.16.0    True        False        False      31m
kube-apiserver                             4.16.0    True        False        False      26m
kube-controller-manager                    4.16.0    True        False        False      36m
kube-scheduler                             4.16.0    True        False        False      36m
kube-storage-version-migrator              4.16.0    True        False        False      37m
machine-api                                4.16.0    True        False        False      29m
machine-approver                           4.16.0    True        False        False      37m
machine-config                             4.16.0    True        False        False      36m
marketplace                                4.16.0    True        False        False      37m
monitoring                                 4.16.0    True        False        False      29m
network                                    4.16.0    True        False        False      38m
node-tuning                                4.16.0    True        False        False      37m
openshift-apiserver                        4.16.0    True        False        False      32m
openshift-controller-manager               4.16.0    True        False        False      30m
openshift-samples                          4.16.0    True        False        False      32m
operator-lifecycle-manager                 4.16.0    True        False        False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False        False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False        False      32m
service-ca                                 4.16.0    True        False        False      38m
storage                                    4.16.0    True        False        False      37m
```

2. 利用不可の Operator を設定します。

関連情報

- OpenShift Container Platform のインストールが失敗した場合にデータを収集する方法の詳細は、失敗したインストールのログの収集 を参照してください。
- クラスター全体で Operator Pod の健全性を確認し、診断用に Operator ログを収集する手順の詳細は、Operator 関連の問題のトラブルシューティング を参照してください。

15.2.15.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

15.2.15.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.2.15.2.1. ベアメタルおよび他の手動インストールの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- ベアメタルなどの、手動でプロビジョニングされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードを使用するクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

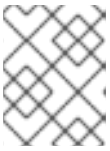
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

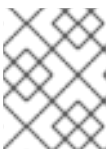
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION              AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16                 True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

15.2.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

15.2.15.2.3. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリューム (または永続ボリューム) はサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

イメージレジストリーでブロックストレージボリュームを使用することを選択した場合は、ファイルシステムの persistent volume claim (PVC) を使用する必要があります。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、1 つの (**1**) レプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
```

```
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
4. 永続ボリューム要求 (PVC) のサイズ。

b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1. カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

15.2.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

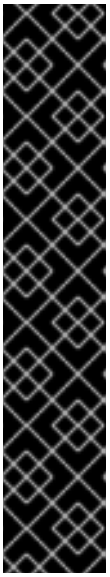
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

15.2.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

15.2.18. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

15.3. ネットワークのカスタマイズを使用したユーザーによってプロビジョニングされるベアメタルクラスターのインストール

OpenShift Container Platform 4.16 では、カスタマイズしたネットワーク設定オプションを使用して、独自にプロビジョニングするベアメタルインフラストラクチャーにクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

OpenShift Container Platform ネットワークをカスタマイズする場合、インストール時にほとんどのネットワーク設定パラメーターを設定する必要があります。実行中のクラスターで変更できるのは **kubeProxy** ネットワーク設定パラメーターのみです。

15.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

15.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブス

クリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

関連情報

- ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーへのネットワークが制限された環境でのインストールの実行についての詳細は、ネットワークが制限された環境でのユーザーによってプロビジョニングされるベアメタルクラスターのインストール を参照してください。

15.3.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

15.3.3.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

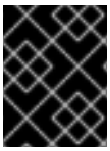
表15.11 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。

ホスト	説明
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

注記

例外として、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターで実行できます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。1 つのコンピュートマシンの実行はサポートされていません。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

15.3.3.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

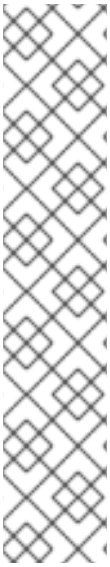
表15.12 最小リソース要件

マシン	オペレーティングシステム	CPU [1]	RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. CPU 1 つ分は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = CPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

15.3.3.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- ベアメタル環境での 3 ノードクラスターのデプロイに関する詳細は、3 ノードクラスターの設定 を参照してください。
- インストール後のクラスター証明書署名要求の承認についての詳細は、マシンの証明書署名要求の承認 を参照してください。

15.3.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

15.3.3.4.1. DHCP を使用したクラスターノードのホスト名の設定

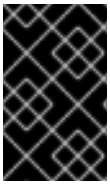
Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

15.3.3.4.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表15.13 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.14 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.15 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定

OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

15.3.3.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.16 必要な DNS レコード

コンポーネント	レコード	説明

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コンピュートマシン	**<compute><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

15.3.3.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例15.4 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
```

```
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例15.5 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
```

```
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

15.3.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

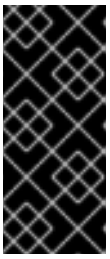
OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.17 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

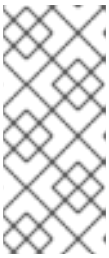
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.18 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

15.3.3.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例15.6 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (5)
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
```

```
listen ingress-router-80 6
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

15.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。

   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

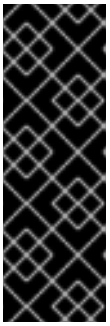
   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。



重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件
- RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始
- DHCP を使用したクラスターノードのホスト名の設定
- 高度な RHCOS インストール設定
- ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件
- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

15.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

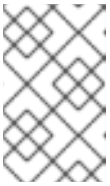
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

関連情報

- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

15.3.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

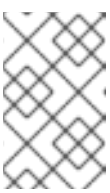
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/.ssh/id_ed25519 などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

関連情報

- ノードの健全性の確認

15.3.7. インストールプログラムの取得

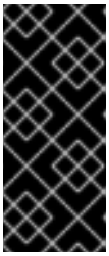
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15.3.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

15.3.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

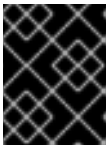
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- ベアメタルのインストール設定パラメーター

15.3.9.1. ベアメタルのサンプル install-config.yaml ファイル

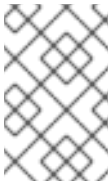
install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OVNKubernetes 11
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths": ...}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



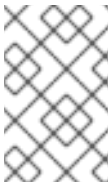
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



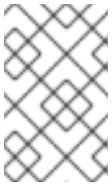
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。プラットフォーム用に追加のプラットフォーム設定変数を指定することはできません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

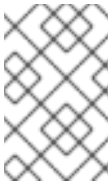
重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

関連情報

- API およびアプリケーションの Ingress 負荷分散要件の詳細は、ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件 を参照してください。

15.3.10. ネットワーク設定フェーズ

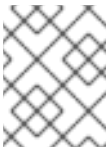
OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**

- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

重要

CIDR 範囲 **172.17.0.0/16** は libVirt によって予約されています。この範囲、またはこの範囲と重複する範囲をクラスター内のネットワークに使用することはできません。

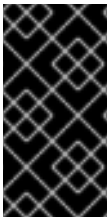
フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 でネットワークプラグインをさらにカスタマイズできます。

15.3.11. 高度なネットワーク設定の指定

ネットワークプラグインに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。



重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1. **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 次の例のように、**cluster-network-03-config.yml** ファイルでクラスターの高度なネットワーク設定を指定します。

   OVN-Kubernetes ネットワークプロバイダーを有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

15.3.12. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

15.3.12.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

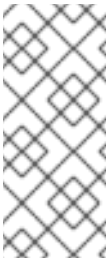
表15.19 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表15.20 defaultNetwork オブジェクト

フィールド	型	説明

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表15.21 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。

フィールド	型	説明
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。  **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表15.22 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。

フィールド	型	説明
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表15.23 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表15.24 **policyAuditConfig オブジェクト**

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。

フィールド	型	説明
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表15.25 **gatewayConfig** オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。

フィールド	型	説明
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表15.26 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表15.27 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表15.28 **ipsecConfig** オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
```

```
type: OVNKubernetes
ovnKubernetesConfig:
  mtu: 1400
  genevePort: 6081
    ipsecConfig:
      mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表15.29 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: `kubeProxyConfig:` `  proxyArguments:` `  iptables-min-sync-period:` `  - 0s`

15.3.13. Ignition 設定ファイルの作成

クラスターマシンは手動で起動する必要があるため、クラスターがマシンを作成するために必要な Ignition 設定ファイルを生成する必要があります。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

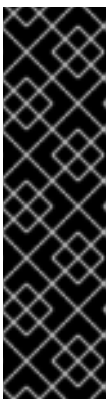
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

- Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

install-config.yaml ファイルを作成している場合、それが含まれるディレクトリーを指定します。または、空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

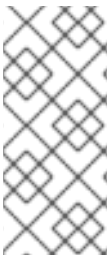
以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

15.3.14. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

RHCOS をマシンにインストールするには、ISO イメージまたはネットワーク PXE ブートを使用する手順のいずれかを実行します。



注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL8 コンピュートマシンのみがサポートされています。

以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブインストール起動プロセスを中断してカーネル引数を追加できます。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。

- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールするノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。この特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。

注記

OpenShift Container Platform 4.6 の時点で、RHCOS ISO およびその他のインストールアーティファクトは、4K セクターのディスクへのインストールをサポートします。

15.3.14.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

   重要

   HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

   出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

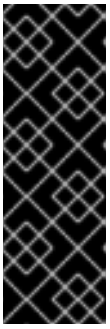
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。

- ディスクに ISO イメージを書き込み、これを直接起動します。
- Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。

6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。

注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

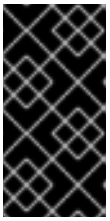
9. RHCOS のインストール後、システムを再起動する必要があります。システムの再起動後、指定した Ignition 設定ファイルを適用します。

10. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

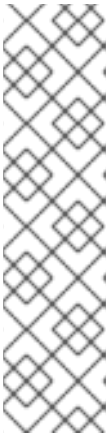
11. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

15.3.14.2. PXE または iPXE ブートを使用した RHCOS のインストール

PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE または iPXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

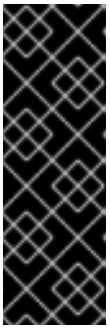
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+
(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```



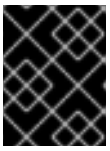
重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs**、**kernel**、および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE または iPXE インストールを設定し、インストールを開始します。ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

- PXE(**x86_64**) の場合:

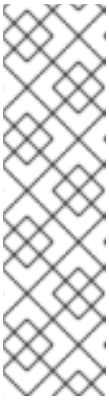
```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> 1
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 2 3
```

1 1 HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

2 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

3 HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値は

ター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

- iPXE (**x86_64** + **aarch64**) の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

(1) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

注記

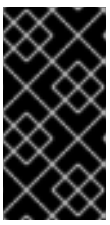
aarch64 アーキテクチャーで CoreOS **kernel** をネットワークブートするには、**IMAGE_GZIP** オプションが有効になっているバージョンの iPXE ビルドを使用する必要があります。iPXE の **IMAGE_GZIP** オプション を参照してください。

- **aarch64** 上の PXE (第 2 段階として UEFI と Grub を使用) の場合:

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
    initrd rhcos-<version>-live-initramfs.<architecture>.img 3
}
```

1. HTTP/TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は、TFTP サーバー上の **kernel** ファイルの場所になります。**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値は HTTP サーバー上のブートストラップ Ignition 設定ファイルの場所になります。
2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。
3. TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

7. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

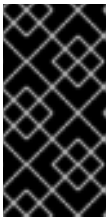
OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。
9. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

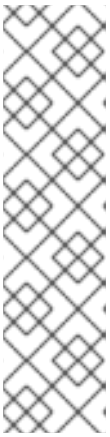
10. クラスターのマシンの作成を続行します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

15.3.14.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。本セクションでは、以下のような手法で実行できるいくつかの設定について説明します。

- カーネル引数をライブインストーラーに渡す
- ライブシステムからの **coreos-installer** の手動による実行
- ライブ ISO または PXE ブートイメージのカスタマイズ

本セクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

15.3.14.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

- 詳細の RHCOS インストールリファレンスの表を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

手順

1. ISO インストーラーを起動します。
2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。
3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```

重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

4. インストール済みのシステムで再起動します。

関連情報

- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

15.3.14.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。デフォルトのパーティション設定をオーバーライドしない限り、特定のアーキテクチャーの各 RHCOS ノードで同じパーティションレイアウトが使用されます。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。

重要

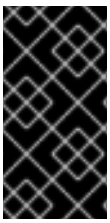
ノードでカスタムパーティションスキームを使用すると、OpenShift Container Platform が一部のノードパーティションでモニタリングやアラートを行わなくなる可能性があります。デフォルトのパーティション設定をオーバーライドする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

OpenShift Container Platform は、次の 2 つのファイルシステム識別子を監視します。

- **nodefs**: **/var/lib/kubelet** を含むファイルシステム
- **imagefs**: **/var/lib/containers** を含むファイルシステム

デフォルトのパーティションスキームの場合、**nodefs** と **imagefs** は同じルートファイルシステム (/) を監視します。

RHCOS を OpenShift Container Platform クラスターノードにインストールするときにデフォルトのパーティション設定をオーバーライドするには、別のパーティションを作成する必要があります。コンテナーとコンテナーイメージ用に別のストレージパーティションを追加する状況を考えてみましょう。たとえば、**/var/lib/containers** を別のパーティションにマウントすると、kubelet が **/var/lib/containers** を **imagefs** ディレクトリーとして、ルートファイルシステムを **nodefs** ディレクトリーとして個別に監視します。



重要

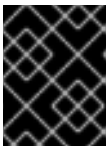
より大きなファイルシステムをホストするためにディスクサイズを変更した場合は、別の **/var/lib/containers** パーティションを作成することを検討してください。多数の割り当てグループによって発生する CPU 時間の問題を軽減するには、**xfs** 形式のディスクのサイズを変更することを検討してください。

15.3.14.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。



重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

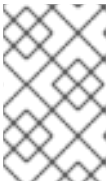
1. インストールホストで、OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

15.3.14.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、**coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。

注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/disk/by-id/scsi-<serial_number>
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/disk/by-id/scsi-<serial_number>
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/disk/by-id/scsi-<serial_number>
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

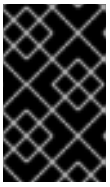
この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

15.3.14.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。

  重要

  これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

  PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **Live install Ignition config**: このタイプは、**coreos-installer customize** サブコマンドとそのさまざまなオプションを使用して作成できます。この方法では、Ignition 設定はライブインストールメディアに渡され、起動直後に実行され、RHCOS システムがディスクにインストールされる前または後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。

PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、 Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

15.3.14.3.4. デフォルトのコンソール設定

OpenShift Container Platform 4.16 ブートイメージからインストールされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードは、ほとんどの仮想化セットアップおよびベアメタルセットアップに対応するためのデフォルトコンソールを使用します。クラウドおよび仮想化プラットフォームが異なれば、選択したアーキテクチャーに応じて、異なるデフォルト設定が使用される場合があります。ベアメタルインストールではカーネルのデフォルト設定が使用されます。これは通常、グラフィカルコンソールがプライマリーコンソールで、シリアルコンソールが無効になっていることを意味します。

デフォルトのコンソールが特定のハードウェア設定と一致しない場合や、デフォルトのコンソールを調整する必要がある特定のニーズがある場合があります。以下に例を示します。

- デバッグ目的で、コンソールの緊急シェルにアクセスしたいと考えています。
- クラウドプラットフォームは、グラフィカルコンソールへの対話型アクセスを提供しませんが、シリアルコンソールを提供します。
- 複数のコンソールを有効にしたい。

コンソール設定は、ブートイメージから継承されます。これは、既存のクラスター内の新しいノードが、デフォルトのコンソールへの変更の影響を受けないことを意味します。

次の方法で、ベアメタルインストール用にコンソールを設定できます。

- コマンドラインで手動で **coreos-installer** を使用する。
- **--dest-console** オプションを指定して **coreos-installer iso Customize** または **coreos-installer pxe Customize** サブコマンドを使用して、プロセスを自動化するカスタムイメージを作成します。

注記

高度なカスタマイズを行うには、カーネル引数ではなく、**coreos-installer iso** または **coreos-installer pxe** サブコマンドを使用してコンソール設定を実行します。

15.3.14.3.5. PXE および ISO インストール用のシリアルコンソールの有効化

デフォルトでは、Red Hat Enterprise Linux CoreOS (RHCOS) シリアルコンソールは無効になっており、すべての出力はグラフィカルコンソールに書き込まれます。ISO インストール用にシリアルコンソールを有効にし、シリアルコンソールとグラフィカルコンソールの両方に出力が送信されるようにブートローダーを再設定できます。

手順

1. ISO インストーラーを起動します。
2. **coreos-installer** コマンドを実行してシステムをインストールし、**--console** オプションを 1 回追加してグラフィカルコンソールを指定し、2 回目にシリアルコンソールを指定します。

```
$ coreos-installer install \
  --console=tty0 \ 1
```

```
--console=ttyS0,<options> \ 2
--ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```

1 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。

2 望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **11520n8** です。オプションが指定されていない場合、デフォルトのカーネル値である **9600n8** が使用されます。このオプションの形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。

3. インストール済みのシステムで再起動します。

注記

coreos-installer install --append-karg オプションを使用し、**console=** でコンソールを指定すると、同様の結果が得られます。ただし、これはカーネルのコンソールのみを設定し、ブートローダーは設定しません。

PXE インストールを設定するには、**coreos.inst.install_dev** カーネルコマンドラインオプションが省略されていることを確認し、シェルプロンプトを使用して、上記の ISO インストール手順を使用して手動で **coreos-installer** を実行します。

15.3.14.3.6. ライブ RHCOS ISO または PXE インストールのカスタマイズ

ライブ ISO イメージまたは PXE 環境を使用して、Ignition 設定ファイルをイメージに直接挿入することで RHCOS をインストールできます。これにより、システムのプロビジョニングに使用できるカスタマイズされたイメージが作成されます。

ISO イメージの場合、これを行うメカニズムは **coreos-installer iso customize** サブコマンドです。これは設定に合わせて **.iso** ファイルを変更します。同様に、PXE 環境のメカニズムは、カスタマイズを含む新しい **initramfs** ファイルを作成する **coreos-installer pxe customize** サブコマンドです。

customize サブコマンドは、他のタイプのカスタマイズも埋め込むことができる汎用ツールです。次のタスクは、より一般的なカスタマイズの例です。

- 企業のセキュリティーポリシーで使う必要がある場合に備えて、カスタム CA 証明書を挿入します。
- カーネル引数を必要とせずにネットワーク設定を設定します。
- 任意のプレインストールおよびポストインストールスクリプトまたはバイナリーを埋め込みます。

15.3.14.3.7. ライブ RHCOS ISO イメージのカスタマイズ

coreos-installer iso customize サブコマンドを使用して、ライブ RHCOS ISO イメージを直接カスタマイズできます。ISO イメージを起動すると、カスタマイズが自動的に適用されます。

この機能を使用して、RHCOS を自動的にインストールするように ISO イメージを設定できます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。

2. RHCOS イメージミラー ページと Ignition 設定ファイルから RHCOS ISO イメージを取得し、次のコマンドを実行して、Ignition 設定を ISO イメージに直接挿入します。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
    --dest-ignition bootstrap.ign \ 1
    --dest-device /dev/disk/by-id/scsi-<serial_number> 2
```

1. **openshift-installer** インストールプログラムから生成される Ignition 設定ファイル。
2. このオプションを指定すると、ISO イメージは自動的にインストールを実行します。それ以外の場合は、イメージはインストール用に設定されたままになりますが、**coreos.inst.install_dev** カーネル引数を指定しない限り、自動的にはインストールされません。

3. オプション: ISO イメージのカスタマイズを削除し、イメージを元の状態に戻すには、次のコマンドを実行します。

```
$ coreos-installer iso reset rhcos-<version>-live.x86_64.iso
```

これで、ライブ ISO イメージを再カスタマイズしたり、元の状態で使用したりできます。

カスタマイズを適用すると、それ以降のすべての RHCOS 起動に影響します。

15.3.14.3.7.1. ライブインストール ISO イメージを変更して、シリアルコンソールを有効化

OpenShift Container Platform 4.12 以降でインストールされたクラスターでは、シリアルコンソールはデフォルトで無効になり、すべての出力がグラフィカルコンソールに書き込まれます。次の手順でシリアルコンソールを有効にできます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。

2. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して ISO イメージをカスタマイズし、シリアルコンソールが出力を受信できるようにします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
  --dest-ignition <path> \ 1
  --dest-console tty0 \ 2
  --dest-console ttyS0,<options> \ 3
  --dest-device /dev/disk/by-id/scsi-<serial_number> 4
```

1. インストールする Ignition 設定の場所。
2. 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。
3. 望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **115200n8** です。オプションが指定されていない場合、デフォルトのカーネル値である

9600n8 が使用されます。このオプションの形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。

4 インストール先として指定されたディスク。このオプションを省略すると、ISO イメージはインストールプログラムを自動的に実行しますが、**coreos.inst.install_dev** カーネル引数も指定しない限り失敗します。

注記

--dest-console オプションは、ライブ ISO システムではなく、インストールされたシステムに影響します。ライブ ISO システムのコンソールを変更するには、**--live-karg-append** オプションを使用し、**console=** でコンソールを指定します。

カスタマイズが適用され、ISO イメージの後続のすべての起動に影響します。

3. オプション: ISO イメージのカスタマイズを削除してイメージを元の状態に戻すには、次のコマンドを実行します。

```
$ coreos-installer iso reset rhcos-<version>-live.x86_64.iso
```

ライブ ISO イメージを再カスタマイズするか、元の状態で使用できるようになりました。

15.3.14.3.7.2. カスタム認証局を使用するようにライブインストール ISO イメージを変更する

customize サブコマンドの **--ignition-ca** フラグを使用して、認証局 (CA) 証明書を Ignition に提供できます。CA 証明書は、インストールの起動時とインストール済みシステムのプロビジョニング時の両方で使用できます。

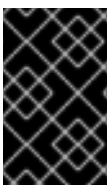
注記

カスタム CA 証明書は、Ignition がリモートリソースをフェッチする方法に影響しますが、システムにインストールされている証明書には影響しません。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して、カスタム CA で使用する ISO イメージをカスタマイズします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso --ignition-ca cert.pem
```



重要

coreos.inst.ignition_url カーネルパラメーターは、**--ignition-ca** フラグでは機能しません。クラスターごとにカスタマイズされたイメージを作成するには、**--dest-ignition** フラグを使用する必要があります。

カスタム CA 証明書を適用すると、それ以降のすべての RHCOS 起動に影響します。

15.3.14.3.7.3. カスタマイズされたネットワーク設定を使用したライブインストール ISO イメージの変更

NetworkManager キーファイルをライブ ISO イメージに埋め込み、**customize** サブコマンドの **--network-keyfile** フラグを使用してインストール済みシステムに渡すことができます。

> **警告**
>
> 接続プロファイルを作成する際は、接続プロファイルのファイル名に **.nmconnection** ファイル名拡張子を使用する必要があります。**.nmconnection** ファイル名拡張子を使用しない場合、クラスターは接続プロファイルをライブ環境に適用しますが、クラスターが初めてノードを起動するときに設定が適用されないため、セットアップが機能しなくなります。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. ボンディングされたインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0.nmconnection** ファイルを作成します。

```
[connection]
id=bond0
type=bond
interface-name=bond0
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto

[proxy]
```

3. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em1.nmconnection** ファイルを作成します。

```
[connection]
id=em1
type=ethernet
interface-name=em1
```

```
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

4. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em2.nmconnection** ファイルを作成します。

```
[connection]
id=em2
type=ethernet
interface-name=em2
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

5. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して、設定されたネットワークで ISO イメージをカスタマイズします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
    --network-keyfile bond0.nmconnection \
    --network-keyfile bond0-proxy-em1.nmconnection \
    --network-keyfile bond0-proxy-em2.nmconnection
```

ネットワーク設定はライブシステムに適用され、宛先システムに引き継がれます。

15.3.14.3.8. ライブ RHCOS PXE 環境のカスタマイズ

coreos-installer pxe customize サブコマンドを使用して、ライブ RHCOS PXE 環境を直接カスタマイズできます。PXE 環境を起動すると、カスタマイズが自動的に適用されます。

この機能を使用して、RHCOS を自動的にインストールするように PXE 環境を設定できます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。

2. RHCOS イメージミラーページと Ignition 設定ファイルから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、Ignition 設定からのカスタマイズを含む新しい **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
    --dest-ignition bootstrap.ign \ 1
    --dest-device /dev/disk/by-id/scsi-<serial_number> \ 2
    -o rhcos-<version>-custom-initramfs.x86_64.img 3
```

1. **openshift-installer** から生成された Ignition 設定ファイル。
2. このオプションを指定すると、PXE 環境で自動的にインストールが実行されます。それ以外の場合は、イメージはインストール用に設定されたままになります が、**coreos.inst.install_dev** カーネル引数を指定しない限り、自動的には設定されません。
3. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。

カスタマイズを適用すると、それ以降のすべての RHCOS 起動に影響します。

15.3.14.3.8.1. ライブインストール PXE 環境を変更して、シリアルコンソールを有効化。

OpenShift Container Platform 4.12 以降でインストールされたクラスターでは、シリアルコンソールはデフォルトで無効になり、すべての出力がグラフィカルコンソールに書き込まれます。次の手順でシリアルコンソールを有効にできます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページおよび Ignition 設定ファイルから RHCOS **kernel**、**initramfs** および **rootfs** ファイルを取得します。次のコマンドを実行して、シリアルコンソールが出力を受信できるようにする新しいカスタマイズされた **initramfs** ファイルを作成します。

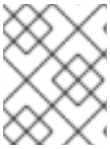
```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
  --dest-ignition <path> \ 1
  --dest-console tty0 \ 2
  --dest-console ttyS0,<options> \ 3
  --dest-device /dev/disk/by-id/scsi-<serial_number> \ 4
  -o rhcos-<version>-custom-initramfs.x86_64.img 5
```

1. インストールする Ignition 設定の場所。
2. 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。
3. 望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **115200n8** です。オプションが指定されていない場合、デフォルトのカーネル値である **9600n8** が使用されます。このオプションの形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。
4. インストール先として指定されたディスク。このオプションを省略すると、PXE 環境は自動的にインストーラーを実行しますが、**coreos.inst.install_dev** カーネル引数も指定しない限り失敗します。
5. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。

カスタマイズが適用され、PXE 環境の後続のすべての起動に影響します。

15.3.14.3.8.2. カスタム認証局を使用するようにライブインストール PXE 環境を変更する

customize サブコマンドの **--ignition-ca** フラグを使用して、認証局 (CA) 証明書を Ignition に提供できます。CA 証明書は、インストールの起動時とインストール済みシステムのプロビジョニング時の両方で使用できます。



注記

カスタム CA 証明書は、Ignition がリモートリソースをフェッチする方法に影響しますが、システムにインストールされている証明書には影響しません。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、カスタム CA で使用するための新しいカスタマイズされた **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
    --ignition-ca cert.pem \
    -o rhcos-<version>-custom-initramfs.x86_64.img
```

3. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。

重要

coreos.inst.ignition_url カーネルパラメーターは、**--ignition-ca** フラグでは機能しません。クラスターごとにカスタマイズされたイメージを作成するには、**--dest-ignition** フラグを使用する必要があります。

カスタム CA 証明書を適用すると、それ以降のすべての RHCOS 起動に影響します。

15.3.14.3.8.3. カスタマイズされたネットワーク設定を使用したライブインストール PXE 環境の変更

NetworkManager キーファイルをライブ PXE 環境に埋め込み、**customize** サブコマンドの **--network-keyfile** フラグを使用して、インストール済みシステムに渡すことができます。

警告

接続プロファイルを作成する際は、接続プロファイルのファイル名に **.nmconnection** ファイル名拡張子を使用する必要があります。**.nmconnection** ファイル名拡張子を使用しない場合、クラスターは接続プロファイルをライブ環境に適用しますが、クラスターが初めてノードを起動するときに設定が適用されないため、セットアップが機能しなくなります。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。

2. ボンディングされたインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0.nmconnection** ファイルを作成します。

```
[connection]
id=bond0
type=bond
interface-name=bond0
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto

[proxy]
```

3. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em1.nmconnection** ファイルを作成します。

```
[connection]
id=em1
type=ethernet
interface-name=em1
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

4. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em2.nmconnection** ファイルを作成します。

```
[connection]
id=em2
type=ethernet
interface-name=em2
master=bond0
multi-connect=1
permissions=
```

```
slave-type=bond

[ethernet]
mac-address-blacklist=
```

5. RHCOS イメージミラー ページから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、設定済みのネットワークを含む新しいカスタマイズされた **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
    --network-keyfile bond0.nmconnection \
    --network-keyfile bond0-proxy-em1.nmconnection \
    --network-keyfile bond0-proxy-em2.nmconnection \
    -o rhcos-<version>-custom-initramfs.x86_64.img
```

6. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。
ネットワーク設定はライブシステムに適用され、宛先システムに引き継がれます。

15.3.14.3.9. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

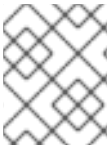
15.3.14.3.9.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定

IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

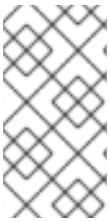
- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1つまたは複数のネットワークを設定する場合、1つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング

オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、options はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

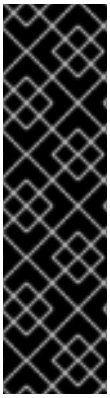
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:em1,em2:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:em1,em2:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合する



重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

オプション: **bond=** オプションを使用して、複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合できます。

各ノードで、次のタスクを実行する必要があります。

1. SR-IOV デバイスの管理 のガイダンスに従って、SR-IOV 仮想機能 (VF) を作成します。「仮想マシンへの SR-IOV ネットワークデバイスの接続」セクションの手順に従います。

2. ボンドを作成し、目的の VF をボンドに接続し、ネットワークボンディングの設定 のガイダンスに従って、ボンドリンクの状態を設定します。説明されている手順のいずれかに従って、結合を作成します。

次の例は、使用する必要がある構文を示しています。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。

<name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は仮想機能 (VF) をカーネル内の既知の名前で表し、**ip link** コマンド (**eno1f0**、**eno2f0**) の出力に表示されます。**options** は結合オプションのコンマ区切りリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、network_interfacesは物理 (イーサネット) インターフェイス (**em1**、**em2**) のコンマ区切りリストを表します。

注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

15.3.14.3.9.2. ISO および PXE インストール用のcoreos-installer オプション

RHCOS は、ISO イメージから RHCOS ライブ環境に起動した後に、コマンドプロンプトで **coreos-installer install <options> <device>** を実行してインストールできます。

以下の表は、**coreos-installer** コマンドに渡すことのできるサブコマンド、オプションおよび引数を示しています。

表15.30 coreos-installer サブコマンド、コマンドラインオプション、および引数

coreos-installer install サブコマンド	
サブコマンド	**説明**

$ coreos-installer install <options> <device>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer install サブコマンドオプション	
オプション	**説明**
-u、--image-url <url>	イメージの URL を手動で指定します。
-f、--image-file <path>	ローカルイメージファイルを手動で指定します。デバッグに使用されます。
-i、--ignition-file <path>	ファイルから Ignition 設定を埋め込みます。
-I、--ignition-url <URL>	URL から Ignition 設定を埋め込みます。
--ignition-hash <digest>	Ignition 設定の **type-value** をダイジェスト値を取得します。
-p、--platform <name>	インストール済みシステムの Ignition プラットフォーム ID を上書きします。
--console <spec>	インストールされたシステムのカーネルとブートローダーコンソールを設定します。**<spec>** の形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。
--append-karg <arg>...	インストール済みシステムにデフォルトのカーネル引数を追加します。
--delete-karg <arg>...	インストール済みシステムからデフォルトのカーネル引数を削除します。
-n、--copy-network	インストール環境からネットワーク設定をコピーします。  **重要** **--copy-network** オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。
--network-dir <path>	**-n** を指定して使用する場合。デフォルトは **/etc/NetworkManager/system-connections/** です。
--save-partlabel <lx>..	このラベル glob でパーティションを保存します。

--save-partindex <id>...	この数または範囲でパーティションを保存します。
--insecure	RHCOS イメージ署名の検証を省略します。
--insecure-ignition	HTTPS またはハッシュなしで Ignition URL を許可します。
--architecture <name>	ターゲット CPU アーキテクチャー。有効な値は **x86_64** および **aarch64** です。
--preserve-on-error	エラー時のパーティションテーブルは消去しないでください。
-h、--help	ヘルプ情報を表示します。
coreos-installer インストールサブコマンド引数	
引数	**説明**
<device>	宛先デバイス。
coreos-installer ISO サブコマンド	
サブコマンド	**説明**
$ coreos-installer iso customize <options> <ISO_image>	RHCOS ライブ ISO イメージをカスタマイズします。
coreos-installer iso reset <options> <ISO_image>	RHCOS ライブ ISO イメージをデフォルト設定に復元します。
coreos-installer iso ignition remove <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を削除します。
coreos-installer ISO カスタマイズサブコマンドオプション	
オプション	**説明**
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。

--dest-karg-append <arg>	宛先システムの各起動にカーネル引数を追加します。
--dest-karg-delete <arg>	宛先システムの各起動からカーネル引数を削除します。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。
--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
--post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
--live-karg-append <arg>	ライブ環境の各ブートにカーネル引数を追加します。
--live-karg-delete <arg>	ライブ環境の各ブートからカーネル引数を削除します。
--live-karg-replace <k=o=n>	ライブ環境の各起動で、**key=old=new** の形式でカーネル引数を置き換えます。
-f, **--force**	既存の Ignition 設定を上書きします。
-o、**--output <path>**	新しい出力ファイルに ISO を書き込みます。
-h、**--help**	ヘルプ情報を表示します。
coreos-installer PXE サブコマンド	
サブコマンド	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	

coreos-installer pxe customize <options> <path>	RHCOS ライブ PXE ブート設定をカスタマイズします。
coreos-installer pxe ignition wrap <options>	イメージに Ignition 設定をラップします。
coreos-installer pxe ignition unwrap <options> <image_name>	イメージでラップされた Ignition 設定を表示します。

coreos-installer PXE はサブコマンドオプションをカスタマイズします

オプション	説明
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。
--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
-o, --output <path>	initramfs を新しい出力ファイルに書き込みます。 **注記** このオプションは、PXE 環境に必要です。

-h、--help	ヘルプ情報を表示します。

15.3.14.3.9.3. ISO または PXE インストールのcoreos.inst ブートオプション

coreos.inst ブートパラメーターを RHCOS ライブインストーラーに渡して、ブート時に **coreos-installer** オプションを自動的に起動できます。これらは、標準のブート引数の追加として提供されます。

- ISO インストールの場合、ブートローダーメニューで自動ブートを中断して **coreos.inst** オプションを追加できます。**RHEL CoreOS (Live)** メニューオプションが強調表示されている状態で **TAB** を押すと、自動ブートを中断できます。
- PXE または iPXE インストールの場合、RHCOS ライブインストーラーのブート前に **coreos.inst** オプションを **APPEND** 行に追加する必要があります。

以下の表は、ISO および PXE インストールの RHCOS ライブインストーラーの **coreos.inst** ブートオプションを示しています。

表15.31coreos.inst ブートオプション

引数	説明
coreos.inst.install_dev	必須。インストール先のシステムのブロックデバイス。**sda** は許可されていますが、**/dev/sda** などの完全パスを使用することが推奨されます。
coreos.inst.ignition_url	オプション: インストール済みシステムに埋め込む Ignition 設定の URL。URL が指定されていない場合、Ignition 設定は埋め込まれません。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.save_partlabel	オプション: インストール時に保存するパーティションのコンマ区切りのラベル。glob 形式のワイルドカードが許可されます。指定したパーティションは存在する必要はありません。
coreos.inst.save_partindex	オプション: インストール時に保存するパーティションのコンマ区切りのインデックス。範囲 **m-n** は許可され、**m** または **n** のいずれかを省略できます。指定したパーティションは存在する必要はありません。
coreos.inst.insecure	オプション: **coreos.inst.image_url** で署名なしと指定される OS イメージを許可します。

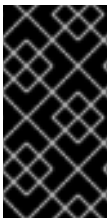
引数	説明
coreos.inst.image_url	オプション: 指定した RHCOS イメージをダウンロードし、インストールします。 ● この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。 ● この引数は、ライブメディアに一致しないバージョンの RHCOS をインストールするために使用できますが、インストールするバージョンに一致するメディアを使用することが推奨されます。 ● **coreos.inst.image_url** を使用している場合は、**coreos.inst.insecure** も使用する必要があります。これは、ベアメタルメディアが OpenShift Container Platform について GPG で署名されていないためです。 ● HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.skip_reboot	オプション: システムはインストール後に再起動しません。インストールが完了するとプロンプトが表示され、インストール時に生じる内容を検査できます。この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。
coreos.inst.platform_id	オプション: RHCOS イメージがインストールされるプラットフォームの Ignition プラットフォーム ID。デフォルトは **metal** です。このオプションは、VMware などのクラウドプロバイダーから Ignition 設定を要求するかどうかを決定します。例: **coreos.inst.platform_id=vmware**
ignition.config.url	オプション: ライブ起動の Ignition 設定の URL。たとえば、これは **coreos-installer** の起動方法をカスタマイズしたり、インストール前後にコードを実行するために使用できます。これはインストール済みシステムの Ignition 設定である **coreos.inst.ignition_url** とは異なります。

15.3.14.4. RHCOS のカーネル引数でのマルチパスの有効化

RHCOS はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。

OpenShift Container Platform 4.8 以降でプロビジョニングされたノードのマルチパスを有効にできます。インストール後のサポートは、マシン設定を使用してマルチパスをアクティベートすることで利用できますが、インストール中にマルチパスを有効にすることが推奨されます。

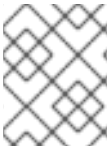
非最適化パスに対して I/O があると、I/O システムエラーが発生するように設定するには、インストール時にマルチパスを有効にする必要があります。



重要

IBM Z® および IBM® LinuxONE では、インストール時にクラスターを設定した場合のみマルチパスを有効にできます。詳細は、**IBM Z® および IBM® LinuxONE への z/VM を使用したクラスターのインストール**の RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始を参照してください。

以下の手順では、インストール時にマルチパスを有効にし、**coreos-installer install** コマンドにカーネル引数を追加して、インストール済みシステム自体が初回起動からマルチパスを使用できるようにします。



注記

OpenShift Container Platform は、4.6 以前からアップグレードされたノードでの day-2 アクティビティーとしてのマルチパスの有効化をサポートしません。

手順

1. マルチパスを有効にして **multipathd** デーモンを起動するには、インストールホストで次のコマンドを実行します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

   - 必要に応じて、PXE または ISO を起動する場合は、カーネルコマンドラインから **rd.multipath=default** を追加することで、マルチパスを有効にできます。

2. **coreos-installer** プログラムを呼び出してカーネル引数を追加します。

   - マシンに接続されているマルチパスデバイスが 1 つしかない場合は、このデバイスは **/dev/mapper/mpatha** のパスで利用できます。以下に例を示します。

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

     1 1 つのマルチパスデバイスのパスを指定します。

   - 複数のマルチパスデバイスがマシンに接続している場合には、より明示的に **/dev/mapper/mpatha** を使用する代わりに、**/dev/disk/by-id** で利用可能な World Wide Name (WWN) シンボリックリンクを使用することが推奨されます。以下に例を示します。

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

     1 マルチパス化されたデバイスの WWN ID を指定します。例: **0xx194e957fcedb4841**

特別な **coreos.inst.*** 引数を使用してライブインストーラーを指定する場合に、このシンボリックリンクを **coreos.inst.install_dev** カーネル引数として使用することもできます。詳細は、Installing RHCOS and starting the OpenShift Container Platform bootstrap process を参照してください。

3. ワーカーノードのいずれかに移動し、カーネルコマンドライン引数 (ホストの **/proc/cmdline** 内) をリスト表示してカーネル引数が機能することを確認します。

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

出力例

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

追加したカーネル引数が表示されるはずです。

15.3.14.5. iSCSI ブートデバイスへの RHCOS の手動インストール

RHCOS を iSCSI ターゲットに手動でインストールできます。

前提条件

1. RHCOS ライブ環境を使用している。
2. RHCOS をインストールする iSCSI ターゲットがある。

手順

1. 次のコマンドを実行して、ライブ環境から iSCSI ターゲットをマウントします。

```
$ iscsiadm \
    --mode discovery \
    --type sendtargets
    --portal <IP_address> \ 1
    --login
```

1 ターゲットポータルの IP アドレス

2. 次のコマンドを実行し、必要なカーネル引数を使用して、RHCOS を iSCSI ターゲットにインストールします。以下に例を示します。

```
$ coreos-installer install \
    /dev/disk/by-path/ip-<IP_address>:<port>-iscsi-<target_iqn>-lun-<lun> \ 1
    --append-karg rd.iscsi.initiator=<initiator_iqn> \ 2
```

```
    --append.karg netroot=<target_iqn> \ 3
    --console ttyS0,115200n8
    --ignition-file <path_to_file>
```

1 インストール先です。ターゲットポータルの IP アドレス、関連するポート番号、ターゲット iSCSI ノードを IQN 形式、および iSCSI 論理ユニット番号(LUN)で指定する必要があります。

2 IQN 形式の iSCSI イニシエーターまたはクライアントの名前。iSCSI イニシエーターは iSCSI ターゲットに接続するセッションを形成します。

3 IQN 形式の iSCSI ターゲットまたはサーバーの名前。

dracut でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

3. 次のコマンドで、iSCSI ディスクをアンマウントします。

```
$ iscsiadm --mode node --logoutall=all
```

この手順は、**coreos-installer iso customize** または **coreos-installer pxe customize** サブコマンドを使用して実行することもできます。

15.3.14.6. iBFT を使用した iSCSI ブートデバイスへの RHCOS のインストール

完全にディスクレスマシンでは、iSCSI ターゲットとイニシエーターの値を iBFT を介して渡すことができます。iSCSI マルチパスもサポートされています。

前提条件

1. RHCOS ライブ環境を使用している。
2. RHCOS をインストールする iSCSI ターゲットがある。
3. オプション：iSCSI ターゲットをマルチパス化している。

手順

1. 次のコマンドを実行して、ライブ環境から iSCSI ターゲットをマウントします。

```
$ iscsiadm \
    --mode discovery \
    --type sendtargets
    --portal <IP_address> \ 1
    --login
```

1 ターゲットポータルの IP アドレス

2. オプション：以下のコマンドでマルチパスを有効にし、デーモンを起動します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

3. 次のコマンドを実行し、必要なカーネル引数を使用して、RHCOS を iSCSI ターゲットにインストールします。以下に例を示します。

```
$ coreos-installer install \
  /dev/mapper/mpatha \ 1
  --append-karg rd.iscsi.firmware=1 \ 2
  --append-karg rd.multipath=default \ 3
  --console ttyS0 \
  --ignition-file <path_to_file>
```

1 1 つのマルチパスデバイスのパス。複数のマルチパスデバイスが接続されている場合、または明示的な場合は、**/dev/disk/by-path** で利用可能な World Wide Name (WWN)シンボリックリンクを使用できます。

2 iSCSI パラメーターは、BIOS ファームウェアから読み込まれます。

3 オプション：マルチパスを有効にする場合は、このパラメーターを含めます。

dracut でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

4. iSCSI ディスクをアンマウントします。

```
$ iscsiadm --mode node --logout=all
```

この手順は、**coreos-installer iso customize** または **coreos-installer pxe customize** サブコマンドを使用して実行することもできます。

15.3.15. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

関連情報

- インストールログの監視と、インストールの問題が発生した場合の診断データの取得に関する詳細は、インストールの進捗の監視 を参照してください。

15.3.16. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.3.17. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

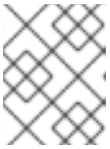
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
master-2 Ready    master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME       AGE     REQUESTOR                                    CONDITION
```

```
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

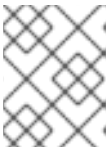
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

15.3.18. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

関連情報

- OpenShift Container Platform のインストールが失敗した場合にデータを収集する方法の詳細は、失敗したインストールのログの収集 を参照してください。
- クラスター全体で Operator Pod の健全性を確認し、診断用に Operator ログを収集する手順の詳細は、Operator 関連の問題のトラブルシューティング を参照してください。

15.3.18.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

15.3.18.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.3.18.3. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリューム (または永続ボリューム) はサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

イメージレジストリーでブロックストレージボリュームを使用することを選択した場合は、ファイルシステムの persistent volume claim (PVC) を使用する必要があります。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、1 つの (**1**) レプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

(1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

(2) **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

(3) 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

(4) 永続ボリューム要求 (PVC) のサイズ。

b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: (1)
```

(1) カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

15.3.19. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。

- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                              1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                              1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                              1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8       1/1
Running    0          5m
...
```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

15.3.20. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

15.3.21. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

15.4. ネットワークが制限された環境でのユーザーによってプロビジョニングされるベアメタルクラスターのインストール

OpenShift Container Platform 4.16 では、制限されたネットワーク内で独自にプロビジョニングするベアメタルインフラストラクチャーに、クラスターをインストールできます。

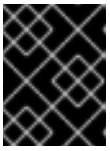
重要

以下の手順に従って仮想化環境またはクラウド環境にクラスターをデプロイすることができますが、ベアメタルプラットフォーム以外の場合は追加の考慮事項に注意してください。このような環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

15.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。

- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターの 永続ストレージ をプロビジョニングした。プライベートイメージレジストリーをデプロイするには、ストレージで ReadWriteMany アクセスモードを指定する必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

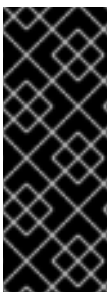
プロキシーを設定する場合は、このサイトリストも確認してください。

15.4.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

15.4.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

15.4.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

15.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

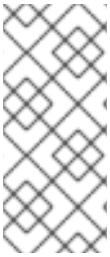
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

15.4.4.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表15.32 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



注記

例外として、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターで実行できます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。1 つのコンピュートマシンの実行はサポートされていません。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

15.4.4.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表15.33 最小リソース要件

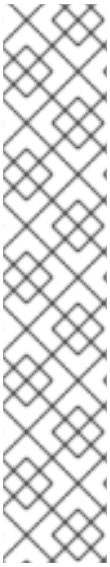
マシン	オペレーティングシステム	CPU [1]	RAM	ストレージ	1 秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. CPU 1 つ分は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = CPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、そ

の他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

15.4.4.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- ベアメタル環境での 3 ノードクラスターのデプロイに関する詳細は、3 ノードクラスターの設定 を参照してください。
- インストール後のクラスター証明書署名要求の承認についての詳細は、マシンの証明書署名要求の承認 を参照してください。

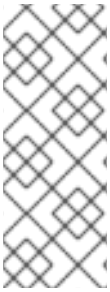
15.4.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その

後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。



注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

15.4.4.4.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

15.4.4.4.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表15.34 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック

プロトコル	ポート	説明
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.35 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.36 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

15.4.4.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

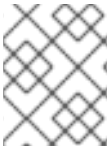
以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.37 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コンピュートマシン	**<compute><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

15.4.4.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例15.7 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
```

```
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例15.8 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
```

```
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
  IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

15.4.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

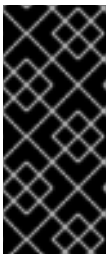
注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



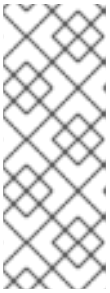
重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.38 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.39 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

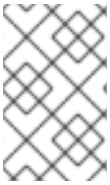
注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

15.4.4.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例15.9 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (5)
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
```

```
listen ingress-router-80 6
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

15.4.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

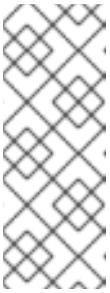
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

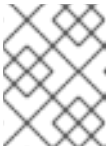
1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。
   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。



注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。



注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件
- RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始
- DHCP を使用したクラスターノードのホスト名の設定
- 高度な RHCOS インストール設定
- ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件
- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

15.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

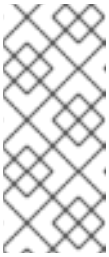
   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```



注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

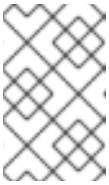
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

関連情報

- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

15.4.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

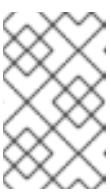
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/.ssh/id_ed25519 などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

関連情報

- ノードの健全性の確認

15.4.8. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

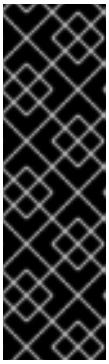
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。
- リポジトリーのミラーリングに使用するコマンドの出力で **imageContentSources** セクションを取得します。
- ミラーレジストリーの証明書の内容を取得する。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

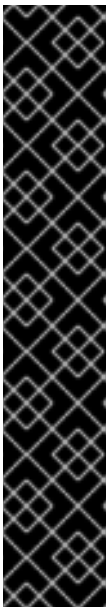
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

- **docker.io** などの、RHCOS がデフォルトで信頼するレジストリーを使用しない限り、**additionalTrustBundle** セクションにミラーリポジトリーの証明書の内容を指定する必要があります。ほとんどの場合、ミラーの証明書を指定する必要があります。
- リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを組み込む必要があります。



重要

- **ImageContentSourcePolicy** ファイルは、ミラーリングプロセスの終了後に **oc mirror** の出力として生成されます。
- **oc mirror** コマンドは、**ImageContentSourcePolicy** の定義に必要な YAML を含む **ImageContentSourcePolicy** ファイルを生成します。このファイルからテキストをコピーし、**install-config.yaml** ファイルに貼り付けます。
- 'oc mirror' コマンドを 2 回実行する必要があります。初めて **oc mirror** コマンドを実行すると、完全な **ImageContentSourcePolicy** ファイルが取得されます。**oc mirror** コマンドを 2 回目に実行すると、1 回目と 2 回目の実行の差のみが得られます。この動作のため、これらのファイルを 1 つの完全な **ImageContentSourcePolicy** ファイルにマージする必要がある場合に備えて、常にこれらのファイルのバックアップを保持する必要があります。これら 2 つの出力ファイルのバックアップを保持すると、完全な **ImageContentSourcePolicy** ファイルが確実に作成されます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- ベアメタルのインストール設定パラメーター

15.4.8.1. ベアメタルのサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

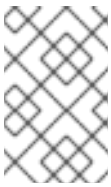
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
```

```
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OVNKubernetes 11
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
additionalTrustBundle: | 17
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 18
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



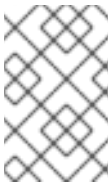
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



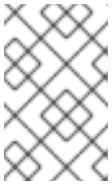
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。プラットフォーム用に追加のプラットフォーム設定変数を指定することはできません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

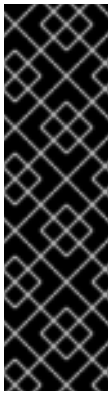
16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

18 リポジトリーのミラーリングに使用したコマンドの出力に従って、**imageContentSources** セクションを指定します。



重要

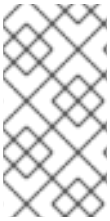
- **oc adm release mirror** コマンドを使用する場合は、**imageContentSources** セクションの出力を使用します。
- **oc mirror** コマンドを使用する場合は、コマンドの実行によって生成される **ImageContentSourcePolicy** ファイルの **repositoryDigestMirrors** セクションを使用します。
- **ImageContentSourcePolicy** は非推奨になりました。詳細は、**イメージレジストリーリポジトリーミラーリングの設定** を参照してください。

関連情報

- API およびアプリケーションの Ingress 負荷分散要件の詳細は、ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件 を参照してください。

15.4.8.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

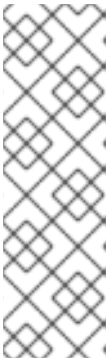


注記

ベアメタルインストールでは、**install-config.yaml** ファイルの **networking.machineNetwork[].cidr** フィールドで指定される範囲にあるノード IP アドレスを割り当てない場合、それらを **proxy.noProxy** フィールドに含める必要があります。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

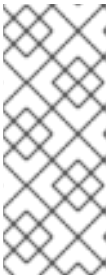
1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

15.4.8.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定されるベアメタルクラスターに、ゼロコンピュートマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの

mastersSchedulable パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

15.4.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- kubelet 証明書のリカバリーに関する詳細は、コントロールプレーン証明書の期限切れの状態からのリカバリー を参照してください。

15.4.10. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定する必要があります。

手順

1. **chrony.conf** ファイルのコンテンツを含む Butane 設定を作成します。たとえば、ワーカーノードで chrony を設定するには、**99-worker-chrony.bu** ファイルを作成します。

注記

Butane の詳細は、"Butane を使用したマシン設定の作成" を参照してください。

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 (3)
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst (4)
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

(1) (2) コントロールプレーンノードでは、これらの両方の場所で **worker** の代わりに **master** を使用します。

(3) マシン設定ファイルの **mode** フィールドに 8 進数の値でモードを指定します。ファイルを作成し、変更を適用すると、**mode** は 10 進数の値に変換されます。コマンド **oc get mc <mc-name> -o yaml** で YAML ファイルを確認できます。

(4) DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-worker-chrony.yaml**) を生成します。

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 以下の 2 つの方法のいずれかで設定を適用します。

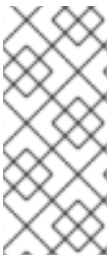
- クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、**MachineConfig** オブジェクトファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を続行します。
- クラスターがすでに実行中の場合は、ファイルを適用します。

```
$ oc apply -f ./99-worker-chrony.yaml
```

15.4.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

RHCOS をマシンにインストールするには、ISO イメージまたはネットワーク PXE ブートを使用する手順のいずれかを実行します。



注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL8 コンピュートマシンのみがサポートされています。

以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブインストール起動プロセスを中断してカーネル引数を追加できます。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。

- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールするノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。この特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。

注記

OpenShift Container Platform 4.6 の時点で、RHCOS ISO およびその他のインストールアーティファクトは、4K セクターのディスクへのインストールをサポートします。

15.4.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

   重要

   HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

   出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

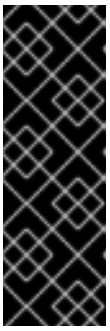
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



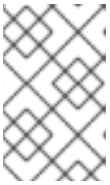
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。



注記

RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。

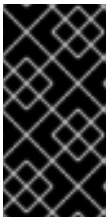
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

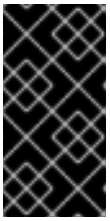
9. RHCOS のインストール後、システムを再起動する必要があります。システムの再起動後、指定した Ignition 設定ファイルを適用します。

10. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

11. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

15.4.11.2. PXE または iPXE ブートを使用した RHCOS のインストール

PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE または iPXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

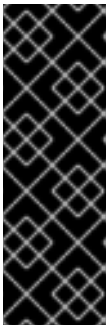
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+
(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```



重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs**、**kernel**、および **initramfs** ファイルを HTTP サーバーにアップロードします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE または iPXE インストールを設定し、インストールを開始します。ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

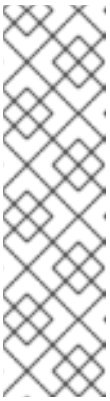
- PXE(**x86_64**) の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パ



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

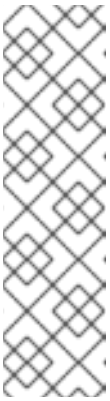
- iPXE (**x86_64** + **aarch64**) の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

(1) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

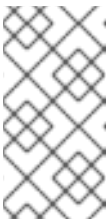
(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。



注記

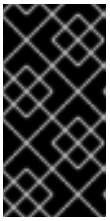
aarch64 アーキテクチャーで CoreOS **kernel** をネットワークブートするには、**IMAGE_GZIP** オプションが有効になっているバージョンの iPXE ビルドを使用する必要があります。iPXE の **IMAGE_GZIP** オプション を参照してください。

- **aarch64** 上の PXE (第 2 段階として UEFI と Grub を使用) の場合:

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
    initrd rhcos-<version>-live-initramfs.<architecture>.img 3
}
```

1. HTTP/TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は、TFTP サーバー上の **kernel** ファイルの場所になります。**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値は HTTP サーバー上のブートストラップ Ignition 設定ファイルの場所になります。
2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。
3. TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

7. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

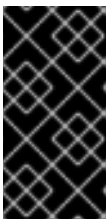
OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。
9. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

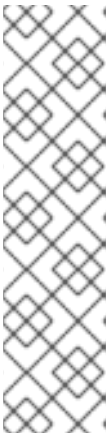
10. クラスターのマシンの作成を続行します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

15.4.11.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。本セクションでは、以下のような手法で実行できるいくつかの設定について説明します。

- カーネル引数をライブインストーラーに渡す
- ライブシステムからの **coreos-installer** の手動による実行
- ライブ ISO または PXE ブートイメージのカスタマイズ

本セクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

15.4.11.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

- 詳細の RHCOS インストールリファレンスの表を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

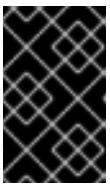
手順

1. ISO インストーラーを起動します。

2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。

3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```



重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

4. インストール済みのシステムで再起動します。

関連情報

- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

15.4.11.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。デフォルトのパーティション設定をオーバーライドしない限り、特定のアーキテクチャーの各 RHCOS ノードで同じパーティションレイアウトが使用されます。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。

重要

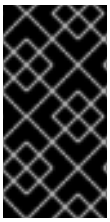
ノードでカスタムパーティションスキームを使用すると、OpenShift Container Platform が一部のノードパーティションでモニタリングやアラートを行わなくなる可能性があります。デフォルトのパーティション設定をオーバーライドする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

OpenShift Container Platform は、次の 2 つのファイルシステム識別子を監視します。

- **nodefs**: **/var/lib/kubelet** を含むファイルシステム
- **imagefs**: **/var/lib/containers** を含むファイルシステム

デフォルトのパーティションスキームの場合、**nodefs** と **imagefs** は同じルートファイルシステム (/) を監視します。

RHCOS を OpenShift Container Platform クラスターノードにインストールするときにデフォルトのパーティション設定をオーバーライドするには、別のパーティションを作成する必要があります。コンテナーとコンテナーイメージ用に別のストレージパーティションを追加する状況を考えてみましょう。たとえば、**/var/lib/containers** を別のパーティションにマウントすると、kubelet が **/var/lib/containers** を **imagefs** ディレクトリーとして、ルートファイルシステムを **nodefs** ディレクトリーとして個別に監視します。



重要

より大きなファイルシステムをホストするためにディスクサイズを変更した場合は、別の **/var/lib/containers** パーティションを作成することを検討してください。多数の割り当てグループによって発生する CPU 時間の問題を軽減するには、**xfs** 形式のディスクのサイズを変更することを検討してください。

15.4.11.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

1. インストールホストで、OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
```

```
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1. **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

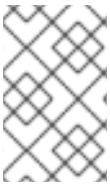
次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

15.4.11.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、**coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。



注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/disk/by-id/scsi-<serial_number>
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/disk/by-id/scsi-<serial_number>
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/disk/by-id/scsi-<serial_number>
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

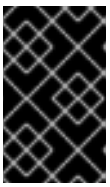
この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

15.4.11.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。

  重要

  これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

  PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **Live install Ignition config**: このタイプは、**coreos-installer customize** サブコマンドとそのさまざまなオプションを使用して作成できます。この方法では、Ignition 設定はライブインストールメディアに渡され、起動直後に実行され、RHCOS システムがディスクにインストールされる前または後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。
  PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、 Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

15.4.11.3.4. デフォルトのコンソール設定

OpenShift Container Platform 4.16 ブートイメージからインストールされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードは、ほとんどの仮想化セットアップおよびベアメタルセットアップに対応す

るためのデフォルトコンソールを使用します。クラウドおよび仮想化プラットフォームが異なれば、選択したアーキテクチャーに応じて、異なるデフォルト設定が使用される場合があります。ベアメタルインストールではカーネルのデフォルト設定が使用されます。これは通常、グラフィカルコンソールがプライマリーコンソールで、シリアルコンソールが無効になっていることを意味します。

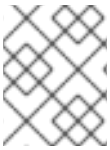
デフォルトのコンソールが特定のハードウェア設定と一致しない場合や、デフォルトのコンソールを調整する必要がある特定のニーズがある場合があります。以下に例を示します。

- デバッグ目的で、コンソールの緊急シェルにアクセスしたいと考えています。
- クラウドプラットフォームは、グラフィカルコンソールへの対話型アクセスを提供しませんが、シリアルコンソールを提供します。
- 複数のコンソールを有効にしたい。

コンソール設定は、ブートイメージから継承されます。これは、既存のクラスター内の新しいノードが、デフォルトのコンソールへの変更の影響を受けないことを意味します。

次の方法で、ベアメタルインストール用にコンソールを設定できます。

- コマンドラインで手動で **coreos-installer** を使用する。
- **--dest-console** オプションを指定して **coreos-installer iso Customize** または **coreos-installer pxe Customize** サブコマンドを使用して、プロセスを自動化するカスタムイメージを作成します。



注記

高度なカスタマイズを行うには、カーネル引数ではなく、**coreos-installer iso** または **coreos-installer pxe** サブコマンドを使用してコンソール設定を実行します。

15.4.11.3.5. PXE および ISO インストール用のシリアルコンソールの有効化

デフォルトでは、Red Hat Enterprise Linux CoreOS (RHCOS) シリアルコンソールは無効になっており、すべての出力はグラフィカルコンソールに書き込まれます。ISO インストール用にシリアルコンソールを有効にし、シリアルコンソールとグラフィカルコンソールの両方に出力が送信されるようにブートローダーを再設定できます。

手順

1. ISO インストーラーを起動します。
2. **coreos-installer** コマンドを実行してシステムをインストールし、**--console** オプションを 1 回追加してグラフィカルコンソールを指定し、2 回目にシリアルコンソールを指定します。

```
$ coreos-installer install \
  --console=tty0 \ 1
  --console=ttyS0,<options> \ 2
  --ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```

1 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。

2

望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **11520n8**

3. インストール済みのシステムで再起動します。

注記

coreos-installer install --append-karg オプションを使用し、**console=** でコンソールを指定すると、同様の結果が得られます。ただし、これはカーネルのコンソールのみを設定し、ブートローダーは設定しません。

PXE インストールを設定するには、**coreos.inst.install_dev** カーネルコマンドラインオプションが省略されていることを確認し、シェルプロンプトを使用して、上記の ISO インストール手順を使用して手動で **coreos-installer** を実行します。

15.4.11.3.6. ライブ RHCOS ISO または PXE インストールのカスタマイズ

ライブ ISO イメージまたは PXE 環境を使用して、Ignition 設定ファイルをイメージに直接挿入することで RHCOS をインストールできます。これにより、システムのプロビジョニングに使用できるカスタマイズされたイメージが作成されます。

ISO イメージの場合、これを行うメカニズムは **coreos-installer iso customize** サブコマンドです。これは設定に合わせて **.iso** ファイルを変更します。同様に、PXE 環境のメカニズムは、カスタマイズを含む新しい **initramfs** ファイルを作成する **coreos-installer pxe customize** サブコマンドです。

customize サブコマンドは、他のタイプのカスタマイズも埋め込むことができる汎用ツールです。次のタスクは、より一般的なカスタマイズの例です。

- 企業のセキュリティーポリシーで使う必要がある場合に備えて、カスタム CA 証明書を挿入します。
- カーネル引数を必要とせずにネットワーク設定を設定します。
- 任意のプレインストールおよびポストインストールスクリプトまたはバイナリーを埋め込みます。

15.4.11.3.7. ライブ RHCOS ISO イメージのカスタマイズ

coreos-installer iso customize サブコマンドを使用して、ライブ RHCOS ISO イメージを直接カスタマイズできます。ISO イメージを起動すると、カスタマイズが自動的に適用されます。

この機能を使用して、RHCOS を自動的にインストールするように ISO イメージを設定できます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページと Ignition 設定ファイルから RHCOS ISO イメージを取得し、次のコマンドを実行して、Ignition 設定を ISO イメージに直接挿入します。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
    --dest-ignition bootstrap.ign \ 1
    --dest-device /dev/disk/by-id/scsi-<serial_number> 2
```

1. **openshift-installer** インストールプログラムから生成される Ignition 設定ファイル。
2. このオプションを指定すると、ISO イメージは自動的にインストールを実行します。それ以外の場合は、イメージはインストール用に設定されたままになります が、**coreos.inst.install_dev** カーネル引数を指定しない限り、自動的にはインストールされません。

3. オプション: ISO イメージのカスタマイズを削除し、イメージを元の状態に戻すには、次のコマンドを実行します。

```
$ coreos-installer iso reset rhcos-<version>-live.x86_64.iso
```

これで、ライブ ISO イメージを再カスタマイズしたり、元の状態で使用したりできます。

カスタマイズを適用すると、それ以降のすべての RHCOS 起動に影響します。

15.4.11.3.7.1. ライブインストール ISO イメージを変更して、シリアルコンソールを有効化

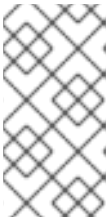
OpenShift Container Platform 4.12 以降でインストールされたクラスターでは、シリアルコンソールはデフォルトで無効になり、すべての出力がグラフィカルコンソールに書き込まれます。次の手順でシリアルコンソールを有効にできます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して ISO イメージをカスタマイズし、シリアルコンソールが出力を受信できるようにします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
  --dest-ignition <path> \ (1)
  --dest-console tty0 \ (2)
  --dest-console ttyS0,<options> \ (3)
  --dest-device /dev/disk/by-id/scsi-<serial_number> (4)
```

1. インストールする Ignition 設定の場所。
2. 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。
3. 望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **115200n8** です。オプションが指定されていない場合、デフォルトのカーネル値である **9600n8** が使用されます。このオプションの形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。
4. インストール先として指定されたディスク。このオプションを省略すると、ISO イメージはインストールプログラムを自動的に実行しますが、**coreos.inst.install_dev** カーネル引数も指定しない限り失敗します。



注記

--dest-console オプションは、ライブ ISO システムではなく、インストールされたシステムに影響します。ライブ ISO システムのコンソールを変更するには、**--live-karg-append** オプションを使用し、**console=** でコンソールを指定します。

カスタマイズが適用され、ISO イメージの後続のすべての起動に影響します。

3. オプション: ISO イメージのカスタマイズを削除してイメージを元の状態に戻すには、次のコマンドを実行します。

```
$ coreos-installer iso reset rhcos-<version>-live.x86_64.iso
```

ライブ ISO イメージを再カスタマイズするか、元の状態で使用できるようになりました。

15.4.11.3.7.2. カスタム認証局を使用するようにライブインストール ISO イメージを変更する

customize サブコマンドの **--ignition-ca** フラグを使用して、認証局 (CA) 証明書を Ignition に提供できます。CA 証明書は、インストールの起動時とインストール済みシステムのプロビジョニング時の両方で使用できます。

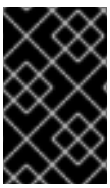
注記

カスタム CA 証明書は、Ignition がリモートリソースをフェッチする方法に影響しますが、システムにインストールされている証明書には影響しません。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して、カスタム CA で使用する ISO イメージをカスタマイズします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso --ignition-ca cert.pem
```



重要

coreos.inst.ignition_url カーネルパラメーターは、**--ignition-ca** フラグでは機能しません。クラスターごとにカスタマイズされたイメージを作成するには、**--dest-ignition** フラグを使用する必要があります。

カスタム CA 証明書を適用すると、それ以降のすべての RHCOS 起動に影響します。

15.4.11.3.7.3. カスタマイズされたネットワーク設定を使用したライブインストール ISO イメージの変更

NetworkManager キーファイルをライブ ISO イメージに埋め込み、**customize** サブコマンドの **--network-keyfile** フラグを使用してインストール済みシステムに渡すことができます。

警告

接続プロファイルを作成する際は、接続プロファイルのファイル名に **.nmconnection** ファイル名拡張子を使用する必要があります。**.nmconnection** ファイル名拡張子を使用しない場合、クラスターは接続プロファイルをライブ環境に適用しますが、クラスターが初めてノードを起動するときに設定が適用されないため、セットアップが機能しなくなります。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。

2. ボンディングされたインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0.nmconnection** ファイルを作成します。

```
[connection]
id=bond0
type=bond
interface-name=bond0
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto

[proxy]
```

3. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em1.nmconnection** ファイルを作成します。

```
[connection]
id=em1
type=ethernet
interface-name=em1
master=bond0
multi-connect=1
permissions=
slave-type=bond
```

```
[ethernet]
mac-address-blacklist=
```

4. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em2.nmconnection** ファイルを作成します。

```
[connection]
id=em2
type=ethernet
interface-name=em2
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

5. RHCOS イメージミラー ページから RHCOS ISO イメージを取得し、次のコマンドを実行して、設定されたネットワークで ISO イメージをカスタマイズします。

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
    --network-keyfile bond0.nmconnection \
    --network-keyfile bond0-proxy-em1.nmconnection \
    --network-keyfile bond0-proxy-em2.nmconnection
```

ネットワーク設定はライブシステムに適用され、宛先システムに引き継がれます。

15.4.11.3.8. ライブ RHCOS PXE 環境のカスタマイズ

coreos-installer pxe customize サブコマンドを使用して、ライブ RHCOS PXE 環境を直接カスタマイズできます。PXE 環境を起動すると、カスタマイズが自動的に適用されます。

この機能を使用して、RHCOS を自動的にインストールするように PXE 環境を設定できます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。

2. RHCOS イメージミラーページと Ignition 設定ファイルから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、Ignition 設定からのカスタマイズを含む新しい **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
    --dest-ignition bootstrap.ign \ 1
    --dest-device /dev/disk/by-id/scsi-<serial_number> \ 2
    -o rhcos-<version>-custom-initramfs.x86_64.img 3
```

1 **openshift-installer** から生成された Ignition 設定ファイル。

2

このオプションを指定すると、PXE 環境で自動的にインストールが実行されます。それ以外の場合は、イメージはインストール用に設定されたままになります

3 PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。

カスタマイズを適用すると、それ以降のすべての RHCOS 起動に影響します。

15.4.11.3.8.1. ライブインストール PXE 環境を変更して、シリアルコンソールを有効化。

OpenShift Container Platform 4.12 以降でインストールされたクラスターでは、シリアルコンソールはデフォルトで無効になり、すべての出力がグラフィカルコンソールに書き込まれます。次の手順でシリアルコンソールを有効にできます。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページおよび Ignition 設定ファイルから RHCOS **kernel**、**initramfs** および **rootfs** ファイルを取得します。次のコマンドを実行して、シリアルコンソールが出力を受信できるようにする新しいカスタマイズされた **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
  --dest-ignition <path> \ 1
  --dest-console tty0 \ 2
  --dest-console ttyS0,<options> \ 3
  --dest-device /dev/disk/by-id/scsi-<serial_number> \ 4
  -o rhcos-<version>-custom-initramfs.x86_64.img 5
```

1 インストールする Ignition 設定の場所。

2 望ましい 2 番目のコンソール。この場合は、グラフィカルコンソールです。このオプションを省略すると、グラフィカルコンソールが無効になります。

3 望ましいひとつ目のコンソール。この場合、シリアルコンソールです。**options** フィールドは、ボーレートとその他の設定を定義します。このフィールドの一般的な値は **115200n8** です。オプションが指定されていない場合、デフォルトのカーネル値である **9600n8** が使用されます。このオプションの形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。

4 インストール先として指定されたディスク。このオプションを省略すると、PXE 環境は自動的にインストーラーを実行しますが、**coreos.inst.install_dev** カーネル引数も指定しない限り失敗します。

5 PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。

カスタマイズが適用され、PXE 環境の後続のすべての起動に影響します。

15.4.11.3.8.2. カスタム認証局を使用するようにライブインストール PXE 環境を変更する

customize サブコマンドの **--ignition-ca** フラグを使用して、認証局 (CA) 証明書を Ignition に提供できます。CA 証明書は、インストールの起動時とインストール済みシステムのプロビジョニング時の両方で使用できます。

注記

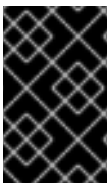
カスタム CA 証明書は、Ignition がリモートリソースをフェッチする方法に影響しますが、システムにインストールされている証明書には影響しません。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。
2. RHCOS イメージミラー ページから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、カスタム CA で使用するための新しいカスタマイズされた **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
    --ignition-ca cert.pem \
    -o rhcos-<version>-custom-initramfs.x86_64.img
```

3. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。



重要

coreos.inst.ignition_url カーネルパラメーターは、**--ignition-ca** フラグでは機能しません。クラスターごとにカスタマイズされたイメージを作成するには、**--dest-ignition** フラグを使用する必要があります。

カスタム CA 証明書を適用すると、それ以降のすべての RHCOS 起動に影響します。

15.4.11.3.8.3. カスタマイズされたネットワーク設定を使用したライブインストール PXE 環境の変更

NetworkManager キーファイルをライブ PXE 環境に埋め込み、**customize** サブコマンドの **--network-keyfile** フラグを使用して、インストール済みシステムに渡すことができます。

警告

接続プロファイルを作成する際は、接続プロファイルのファイル名に **.nmconnection** ファイル名拡張子を使用する必要があります。**.nmconnection** ファイル名拡張子を使用しない場合、クラスターは接続プロファイルをライブ環境に適用しますが、クラスターが初めてノードを起動するときに設定が適用されないため、セットアップが機能しなくなります。

手順

1. **coreos-installer** イメージミラー ページから、**coreos-installer** バイナリーをダウンロードします。

2. ボンディングされたインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0.nmconnection** ファイルを作成します。

```
[connection]
id=bond0
type=bond
interface-name=bond0
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto

[proxy]
```

3. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em1.nmconnection** ファイルを作成します。

```
[connection]
id=em1
type=ethernet
interface-name=em1
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

4. ボンディングに追加するセカンダリーインターフェイスの接続プロファイルを作成します。たとえば、ローカルディレクトリーに次の内容の **bond0-proxy-em2.nmconnection** ファイルを作成します。

```
[connection]
id=em2
type=ethernet
interface-name=em2
master=bond0
multi-connect=1
permissions=
```

```
slave-type=bond

[ethernet]
mac-address-blacklist=
```

5. RHCOS イメージミラー ページから、RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得し、次のコマンドを実行して、設定済みのネットワークを含む新しいカスタマイズされた **initramfs** ファイルを作成します。

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
    --network-keyfile bond0.nmconnection \
    --network-keyfile bond0-proxy-em1.nmconnection \
    --network-keyfile bond0-proxy-em2.nmconnection \
    -o rhcos-<version>-custom-initramfs.x86_64.img
```

6. PXE 設定でカスタマイズされた **initramfs** ファイルを使用します。**ignition.firstboot** および **ignition.platform.id=metal** カーネル引数が存在しない場合は追加します。
ネットワーク設定はライブシステムに適用され、宛先システムに引き継がれます。

15.4.11.3.9. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

15.4.11.3.9.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

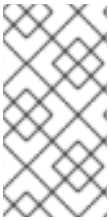
次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定

IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

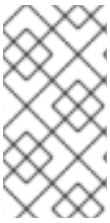
- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング

オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、options はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

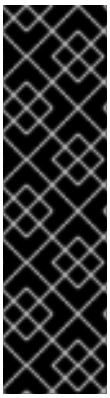
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:em1,em2:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:em1,em2:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合する



重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

オプション: **bond=** オプションを使用して、複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合できます。

各ノードで、次のタスクを実行する必要があります。

1. SR-IOV デバイスの管理 のガイダンスに従って、SR-IOV 仮想機能 (VF) を作成します。「仮想マシンへの SR-IOV ネットワークデバイスの接続」セクションの手順に従います。

2. ボンドを作成し、目的の VF をボンドに接続し、ネットワークボンディングの設定 のガイダンスに従って、ボンドリンクの状態を設定します。説明されている手順のいずれかに従って、結合を作成します。

次の例は、使用する必要がある構文を示しています。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。

<name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は仮想機能 (VF) をカーネル内の既知の名前で表し、**ip link** コマンド (**eno1f0**、**eno2f0**) の出力に表示されます。**options** は結合オプションのコンマ区切りリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

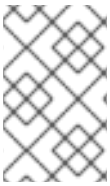
    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、network_interfacesは物理 (イーサネット) インターフェイス (**em1**、**em2**) のコンマ区切りリストを表します。



注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

15.4.11.3.9.2. ISO および PXE インストール用のcoreos-installer オプション

RHCOS は、ISO イメージから RHCOS ライブ環境に起動した後に、コマンドプロンプトで **coreos-installer install <options> <device>** を実行してインストールできます。

以下の表は、**coreos-installer** コマンドに渡すことのできるサブコマンド、オプションおよび引数を示しています。

表15.40 coreos-installer サブコマンド、コマンドラインオプション、および引数

coreos-installer install サブコマンド	
サブコマンド	**説明**

$ coreos-installer install <options> <device>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer install サブコマンドオプション	
オプション	**説明**
-u、--image-url <url>	イメージの URL を手動で指定します。
-f、--image-file <path>	ローカルイメージファイルを手動で指定します。デバッグに使用されます。
-i、--ignition-file <path>	ファイルから Ignition 設定を埋め込みます。
-I、--ignition-url <URL>	URL から Ignition 設定を埋め込みます。
--ignition-hash <digest>	Ignition 設定の **type-value** をダイジェスト値を取得します。
-p、--platform <name>	インストール済みシステムの Ignition プラットフォーム ID を上書きします。
--console <spec>	インストールされたシステムのカーネルとブートローダーコンソールを設定します。**<spec>** の形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。
--append-karg <arg>...	インストール済みシステムにデフォルトのカーネル引数を追加します。
--delete-karg <arg>...	インストール済みシステムからデフォルトのカーネル引数を削除します。
-n、--copy-network	インストール環境からネットワーク設定をコピーします。 **重要** **--copy-network** オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。
--network-dir <path>	**-n** を指定して使用する場合。デフォルトは **/etc/NetworkManager/system-connections/** です。
--save-partlabel <lx>..	このラベル glob でパーティションを保存します。

--save-partindex <id>...	この数または範囲でパーティションを保存します。
--insecure	RHCOS イメージ署名の検証を省略します。
--insecure-ignition	HTTPS またはハッシュなしで Ignition URL を許可します。
--architecture <name>	ターゲット CPU アーキテクチャー。有効な値は **x86_64** および **aarch64** です。
--preserve-on-error	エラー時のパーティションテーブルは消去しないでください。
-h、--help	ヘルプ情報を表示します。
coreos-installer **インストールサブコマンド引数**	
引数	**説明**
<device>	宛先デバイス。
coreos-installer **ISO サブコマンド**	
サブコマンド	**説明**
$ coreos-installer iso customize <options> <ISO_image>	RHCOS ライブ ISO イメージをカスタマイズします。
coreos-installer iso reset <options> <ISO_image>	RHCOS ライブ ISO イメージをデフォルト設定に復元します。
coreos-installer iso ignition remove <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を削除します。
coreos-installer **ISO カスタマイズサブコマンドオプション**	
オプション	**説明**
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。

--dest-karg-append <arg>	宛先システムの各起動にカーネル引数を追加します。
--dest-karg-delete <arg>	宛先システムの各起動からカーネル引数を削除します。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。
--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
--post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
--live-karg-append <arg>	ライブ環境の各ブートにカーネル引数を追加します。
--live-karg-delete <arg>	ライブ環境の各ブートからカーネル引数を削除します。
--live-karg-replace <k=o=n>	ライブ環境の各起動で、**key=old=new** の形式でカーネル引数を置き換えます。
-f, **--force**	既存の Ignition 設定を上書きします。
-o、**--output <path>**	新しい出力ファイルに ISO を書き込みます。
-h、**--help**	ヘルプ情報を表示します。
coreos-installer PXE サブコマンド	
サブコマンド	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	

coreos-installer pxe customize <options> <path>	RHCOS ライブ PXE ブート設定をカスタマイズします。
coreos-installer pxe ignition wrap <options>	イメージに Ignition 設定をラップします。
coreos-installer pxe ignition unwrap <options> <image_name>	イメージでラップされた Ignition 設定を表示します。

coreos-installer PXE はサブコマンドオプションをカスタマイズします

オプション	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。
--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
-o, --output <path>	initramfs を新しい出力ファイルに書き込みます。  **注記** このオプションは、PXE 環境に必要です。

-h、--help	ヘルプ情報を表示します。

15.4.11.3.9.3. ISO または PXE インストールのcoreos.inst ブートオプション

coreos.inst ブートパラメーターを RHCOS ライブインストーラーに渡して、ブート時に **coreos-installer** オプションを自動的に起動できます。これらは、標準のブート引数の追加として提供されます。

- ISO インストールの場合、ブートローダーメニューで自動ブートを中断して **coreos.inst** オプションを追加できます。**RHEL CoreOS (Live)** メニューオプションが強調表示されている状態で **TAB** を押すと、自動ブートを中断できます。
- PXE または iPXE インストールの場合、RHCOS ライブインストーラーのブート前に **coreos.inst** オプションを **APPEND** 行に追加する必要があります。

以下の表は、ISO および PXE インストールの RHCOS ライブインストーラーの **coreos.inst** ブートオプションを示しています。

表15.41 coreos.inst ブートオプション

引数	説明
coreos.inst.install_dev	必須。インストール先のシステムのブロックデバイス。**sda** は許可されていますが、**/dev/sda** などの完全パスを使用することが推奨されます。
coreos.inst.ignition_url	オプション: インストール済みシステムに埋め込む Ignition 設定の URL。URL が指定されていない場合、Ignition 設定は埋め込まれません。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.save_partlabel	オプション: インストール時に保存するパーティションのコンマ区切りのラベル。glob 形式のワイルドカードが許可されます。指定したパーティションは存在する必要はありません。
coreos.inst.save_partindex	オプション: インストール時に保存するパーティションのコンマ区切りのインデックス。範囲 **m-n** は許可され、**m** または **n** のいずれかを省略できます。指定したパーティションは存在する必要はありません。
coreos.inst.insecure	オプション: **coreos.inst.image_url** で署名なしと指定される OS イメージを許可します。

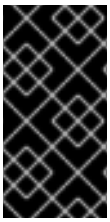
引数	説明
coreos.inst.image_url	オプション: 指定した RHCOS イメージをダウンロードし、インストールします。 ● この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。 ● この引数は、ライブメディアに一致しないバージョンの RHCOS をインストールするために使用できますが、インストールするバージョンに一致するメディアを使用することが推奨されます。 ● **coreos.inst.image_url** を使用している場合は、**coreos.inst.insecure** も使用する必要があります。これは、ベアメタルメディアが OpenShift Container Platform について GPG で署名されていないためです。 ● HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.skip_reboot	オプション: システムはインストール後に再起動しません。インストールが完了するとプロンプトが表示され、インストール時に生じる内容を検査できます。この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。
coreos.inst.platform_id	オプション: RHCOS イメージがインストールされるプラットフォームの Ignition プラットフォーム ID。デフォルトは **metal** です。このオプションは、VMware などのクラウドプロバイダーから Ignition 設定を要求するかどうかを決定します。例: **coreos.inst.platform_id=vmware**
ignition.config.url	オプション: ライブ起動の Ignition 設定の URL。たとえば、これは **coreos-installer** の起動方法をカスタマイズしたり、インストール前後にコードを実行するために使用できます。これはインストール済みシステムの Ignition 設定である **coreos.inst.ignition_url** とは異なります。

15.4.11.4. RHCOS のカーネル引数でのマルチパスの有効化

RHCOS はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。

OpenShift Container Platform 4.8 以降でプロビジョニングされたノードのマルチパスを有効にできます。インストール後のサポートは、マシン設定を使用してマルチパスをアクティベートすることで利用できますが、インストール中にマルチパスを有効にすることが推奨されます。

非最適化パスに対して I/O があると、I/O システムエラーが発生するように設定するには、インストール時にマルチパスを有効にする必要があります。



重要

IBM Z® および IBM® LinuxONE では、インストール時にクラスターを設定した場合のみマルチパスを有効にできます。詳細は、**IBM Z® および IBM® LinuxONE への z/VM を使用したクラスターのインストール**の RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始を参照してください。

以下の手順では、インストール時にマルチパスを有効にし、**coreos-installer install** コマンドにカーネル引数を追加して、インストール済みシステム自体が初回起動からマルチパスを使用できるようにします。

注記

OpenShift Container Platform は、4.6 以前からアップグレードされたノードでの day-2 アクティビティーとしてのマルチパスの有効化をサポートしません。

手順

1. マルチパスを有効にして **multipathd** デーモンを起動するには、インストールホストで次のコマンドを実行します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

   - 必要に応じて、PXE または ISO を起動する場合は、カーネルコマンドラインから **rd.multipath=default** を追加することで、マルチパスを有効にできます。

2. **coreos-installer** プログラムを呼び出してカーネル引数を追加します。

   - マシンに接続されているマルチパスデバイスが 1 つしかない場合は、このデバイスは **/dev/mapper/mpatha** のパスで利用できます。以下に例を示します。

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

     1 1 つのマルチパスデバイスのパスを指定します。

   - 複数のマルチパスデバイスがマシンに接続している場合には、より明示的に **/dev/mapper/mpatha** を使用する代わりに、**/dev/disk/by-id** で利用可能な World Wide Name (WWN) シンボリックリンクを使用することが推奨されます。以下に例を示します。

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

     1 マルチパス化されたデバイスの WWN ID を指定します。例: **0xx194e957fcedb4841**

特別な **coreos.inst.*** 引数を使用してライブインストーラーを指定する場合に、このシンボリックリンクを **coreos.inst.install_dev** カーネル引数として使用することもできます。詳細は、Installing RHCOS and starting the OpenShift Container Platform bootstrap process を参照してください。

3. ワーカーノードのいずれかに移動し、カーネルコマンドライン引数 (ホストの **/proc/cmdline** 内) をリスト表示してカーネル引数が機能することを確認します。

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

出力例

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

追加したカーネル引数が表示されるはずです。

15.4.11.5. iSCSI ブートデバイスへの RHCOS の手動インストール

RHCOS を iSCSI ターゲットに手動でインストールできます。

前提条件

1. RHCOS ライブ環境を使用している。
2. RHCOS をインストールする iSCSI ターゲットがある。

手順

1. 次のコマンドを実行して、ライブ環境から iSCSI ターゲットをマウントします。

```
$ iscsiadm \
    --mode discovery \
    --type sendtargets
    --portal <IP_address> \ 1
    --login
```

1 ターゲットポータルの IP アドレス

2. 次のコマンドを実行し、必要なカーネル引数を使用して、RHCOS を iSCSI ターゲットにインストールします。以下に例を示します。

```
$ coreos-installer install \
    /dev/disk/by-path/ip-<IP_address>:<port>-iscsi-<target_iqn>-lun-<lun> \ 1
    --append-karg rd.iscsi.initiator=<initiator_iqn> \ 2
```

```
    --append.karg netroot=<target_iqn> \ 3
    --console ttyS0,115200n8
    --ignition-file <path_to_file>
```

1 インストール先です。ターゲットポータルの IP アドレス、関連するポート番号、ターゲット iSCSI ノードを IQN 形式、および iSCSI 論理ユニット番号(LUN)で指定する必要があります。

2 IQN 形式の iSCSI イニシエーターまたはクライアントの名前。iSCSI イニシエーターは iSCSI ターゲットに接続するセッションを形成します。

3 IQN 形式の iSCSI ターゲットまたはサーバーの名前。

dracut でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

3. 次のコマンドで、iSCSI ディスクをアンマウントします。

```
$ iscsiadm --mode node --logoutall=all
```

この手順は、**coreos-installer iso customize** または **coreos-installer pxe customize** サブコマンドを使用して実行することもできます。

15.4.11.6. iBFT を使用した iSCSI ブートデバイスへの RHCOS のインストール

完全にディスクレスマシンでは、iSCSI ターゲットとイニシエーターの値を iBFT を介して渡すことができます。iSCSI マルチパスもサポートされています。

前提条件

1. RHCOS ライブ環境を使用している。
2. RHCOS をインストールする iSCSI ターゲットがある。
3. オプション：iSCSI ターゲットをマルチパス化している。

手順

1. 次のコマンドを実行して、ライブ環境から iSCSI ターゲットをマウントします。

```
$ iscsiadm \
    --mode discovery \
    --type sendtargets
    --portal <IP_address> \ 1
    --login
```

1 ターゲットポータルの IP アドレス

2. オプション：以下のコマンドでマルチパスを有効にし、デーモンを起動します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

3. 次のコマンドを実行し、必要なカーネル引数を使用して、RHCOS を iSCSI ターゲットにインストールします。以下に例を示します。

```
$ coreos-installer install \
  /dev/mapper/mpatha \ 1
  --append-karg rd.iscsi.firmware=1 \ 2
  --append-karg rd.multipath=default \ 3
  --console ttyS0 \
  --ignition-file <path_to_file>
```

1. 1 つのマルチパスデバイスのパス。複数のマルチパスデバイスが接続されている場合、または明示的な場合は、**/dev/disk/by-path** で利用可能な World Wide Name (WWN)シンボリックリンクを使用できます。
2. iSCSI パラメーターは、BIOS ファームウェアから読み込まれます。
3. オプション：マルチパスを有効にする場合は、このパラメーターを含めます。

dracut でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

4. iSCSI ディスクをアンマウントします。

```
$ iscsiadm --mode node --logout=all
```

この手順は、**coreos-installer iso customize** または **coreos-installer pxe customize** サブコマンドを使用して実行することもできます。

15.4.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

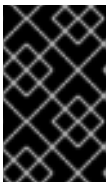
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

関連情報

- インストールログの監視と、インストールの問題が発生した場合の診断データの取得に関する詳細は、インストールの進捗の監視 を参照してください。

15.4.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.4.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

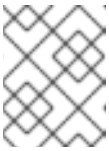
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal        Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal        Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

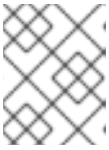
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME       STATUS   ROLES    AGE  VERSION
   master-0   Ready    master   73m  v1.29.4
   master-1   Ready    master   73m  v1.29.4
   master-2   Ready    master   74m  v1.29.4
   worker-0   Ready    worker   11m  v1.29.4
   worker-1   Ready    worker   11m  v1.29.4
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

15.4.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

関連情報

- OpenShift Container Platform のインストールが失敗した場合にデータを収集する方法の詳細は、失敗したインストールのログの収集 を参照してください。
- クラスター全体で Operator Pod の健全性を確認し、診断用に Operator ログを収集する手順の詳細は、Operator 関連の問題のトラブルシューティング を参照してください。

15.4.15.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

15.4.15.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.4.15.2.1. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、Image Registry Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **managementState** Image Registry Operator 設定を **Removed** から **Managed** に変更します。以下に例を示します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

15.4.15.2.2. ベアメタルおよび他の手動インストールの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- ベアメタルなどの、手動でプロビジョニングされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードを使用するクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

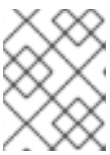
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION       AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16          True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

15.4.15.2.3. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

15.4.15.2.4. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリューム (または永続ボリューム) はサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

イメージレジストリーでブロックストレージボリュームを使用することを選択した場合は、ファイルシステムの persistent volume claim (PVC) を使用する必要があります。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、1 つの (**1**) レプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1 **PersistentVolumeClaim** オブジェクトを表す一意の名前。

2 **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

3

永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

4 永続ボリューム要求 (PVC) のサイズ。

b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

15.4.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                            4.16.0    True        False         False      40m
cluster-autoscaler                          4.16.0    True        False         False      37m
config-operator                             4.16.0    True        False         False      38m
```

```
console                                    4.16.0    True        False        False      26m
csi-snapshot-controller                    4.16.0    True        False        False      37m
dns                                        4.16.0    True        False        False      37m
etcd                                       4.16.0    True        False        False      36m
image-registry                             4.16.0    True        False        False      31m
ingress                                    4.16.0    True        False        False      30m
insights                                   4.16.0    True        False        False      31m
kube-apiserver                             4.16.0    True        False        False      26m
kube-controller-manager                    4.16.0    True        False        False      36m
kube-scheduler                             4.16.0    True        False        False      36m
kube-storage-version-migrator              4.16.0    True        False        False      37m
machine-api                                4.16.0    True        False        False      29m
machine-approver                           4.16.0    True        False        False      37m
machine-config                             4.16.0    True        False        False      36m
marketplace                                4.16.0    True        False        False      37m
monitoring                                 4.16.0    True        False        False      29m
network                                    4.16.0    True        False        False      38m
node-tuning                                4.16.0    True        False        False      37m
openshift-apiserver                        4.16.0    True        False        False      32m
openshift-controller-manager               4.16.0    True        False        False      30m
openshift-samples                          4.16.0    True        False        False      32m
operator-lifecycle-manager                 4.16.0    True        False        False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False        False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False        False      32m
service-ca                                 4.16.0    True        False        False      38m
storage                                    4.16.0    True        False        False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

4. Cluster registration ページでクラスターを登録します。

15.4.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

15.4.18. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズ します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。

15.5. BARE METAL OPERATOR を使用したユーザープロビジョニングクラスターのスケーリング

ユーザーがプロビジョニングしたインフラストラクチャークラスターをデプロイした後、Bare Metal Operator (BMO) およびその他の metal3 コンポーネントを使用して、クラスター内のベアメタルホストをスケーリングできます。このアプローチは、ユーザーがプロビジョニングしたクラスターをより自動化された方法でスケーリングするために役立ちます。

15.5.1. Bare Metal Operator を使用したユーザープロビジョニングクラスターのスケーリングについて

Bare Metal Operator (BMO) およびその他の metal3 コンポーネントを使用して、ユーザーがプロビジョニングしたインフラストラクチャークラスターをスケーリングできます。ユーザーがプロビジョニングしたインフラストラクチャーのインストールには、Machine API Operator が含まれていません。Machine API Operator は通常、クラスター内のベアメタルホストのライフサイクルを管理します。ただし、マシン API オペレーターを必要とせずに、BMO およびその他の metal3 コンポーネントを使用して、ユーザーがプロビジョニングしたクラスター内のノードをスケーリングすることができます。

15.5.1.1. ユーザーがプロビジョニングしたクラスターをスケーリングするための前提条件

- ユーザーがプロビジョニングしたインフラストラクチャークラスターをベアメタルにインストールしました。
- ホストへのベースボード管理コントローラー (BMC) アクセス権限がある。

15.5.1.2. ユーザーがプロビジョニングしたクラスターのスケーリングに関する制限事項

- Bare Metal Operator (BMO) では、プロビジョニングネットワークを使用して、ユーザーがプロビジョニングしたインフラストラクチャークラスターをスケーリングすることはできません。
  - したがって、仮想メディアネットワークの起動をサポートするベアメタルホストドライバー (**redfish-virtualmedia** や **idrac-virtualmedia** など) のみを使用できます。
- BMO を使用して、ユーザーがプロビジョニングしたインフラストラクチャークラスター内の **MachineSet** オブジェクトをスケーリングすることはできません。

15.5.2. ユーザーがプロビジョニングしたクラスターをスケーリングするためのプロビジョニングリソースの設定

Provisioning カスタムリソース (CR) を作成して、ユーザーがプロビジョニングしたインフラストラクチャークラスターで Metal プラットフォームコンポーネントを有効にします。

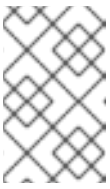
前提条件

- ユーザーがプロビジョニングしたインフラストラクチャークラスターをベアメタルにインストールしました。

手順

1. **Provisioning** CR を作成します。
   a. 次の YAML を **provisioning.yaml** ファイルに保存します。

```
apiVersion: metal3.io/v1alpha1
kind: Provisioning
metadata:
  name: provisioning-configuration
spec:
  provisioningNetwork: "Disabled"
  watchAllNamespaces: false
```



注記

OpenShift Container Platform 4.16 では、Bare Metal Operator を使用してユーザーがプロビジョニングしたクラスターをスケーリングする場合、プロビジョニングネットワークの有効化がサポートされません。

2. 次のコマンドを実行して、**Provisioning** CR を作成します。

```
$ oc create -f provisioning.yaml
```

出力例

```
provisioning.metal3.io/provisioning-configuration created
```

検証

- 次のコマンドを実行して、プロビジョニングサービスが実行されていることを確認します。

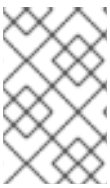
```
$ oc get pods -n openshift-machine-api
```

出力例

```
NAME                                           READY   STATUS    RESTARTS       AGE
cluster-autoscaler-operator-678c476f4c-jjdn5   2/2     Running   0              5d21h
cluster-baremetal-operator-6866f7b976-gmvgh    2/2     Running   0              5d21h
control-plane-machine-set-operator-7d8566696c-bh4jz   1/1     Running   0              5d21h
ironic-proxy-64bdw                             1/1     Running   0              5d21h
ironic-proxy-rbggf                             1/1     Running   0              5d21h
ironic-proxy-vj54c                             1/1     Running   0              5d21h
machine-api-controllers-544d6849d5-tgj9l       7/7     Running   1 (5d21h ago)   5d21h
machine-api-operator-5c4ff4b86d-6fjmq          2/2     Running   0              5d21h
metal3-6d98f84cc8-zn2mx                        5/5     Running   0              5d21h
metal3-image-customization-59d745768d-bhrp7    1/1     Running   0              5d21h
```

15.5.3. BMO を使用して、ユーザーがプロビジョニングしたクラスターで新しいホストをプロビジョニングする

Bare Metal Operator (BMO) を使用して、**BareMetalHost** カスタムリソース (CR) を作成すると、ユーザーがプロビジョニングしたクラスターにベアメタルホストをプロビジョニングできます。



注記

BMO を使用して、クラスターにベアメタルホストをプロビジョニングするには、**BareMetalHost** カスタムリソースの **spec.externallyProvisioned** 仕様を **false** に設定する必要があります。

前提条件

- ユーザーがプロビジョニングしたベアメタルクラスターを作成しました。
- ホストへのベースボード管理コントローラー (BMC) アクセス権限がある。
- **Provisioning** CR を作成して、クラスターにプロビジョニングサービスをデプロイしました。

手順

1. **Secret** CR と **BareMetalHost** CR を作成します。
   a. 次の YAML を **bmh.yaml** ファイルに保存します。

```
---
apiVersion: v1
kind: Secret
```

```
metadata:
  name: worker1-bmc
  namespace: openshift-machine-api
type: Opaque
data:
  username: <base64_of_uid>
  password: <base64_of_pwd>
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: worker1
  namespace: openshift-machine-api
spec:
  bmc:
    address: <protocol>://<bmc_url> (1)
    credentialsName: "worker1-bmc"
  bootMACAddress: <nic1_mac_address>
  externallyProvisioned: false (2)
  customDeploy:
    method: install_coreos
  online: true
  userData:
    name: worker-user-data-managed
    namespace: openshift-machine-api
```

(1) 仮想メディアネットワークの起動をサポートするベアメタルホストドライバー (**redfish-virtualmedia** や **idrac-virtualmedia** など) のみを使用できます。

(2) **BareMetalHost** カスタムリソースの **spec.externallyProvisioned** 仕様を **false** に設定する必要があります。デフォルト値は **false** です。

2. 次のコマンドを実行して、ベアメタルホストオブジェクトを作成します。

```
$ oc create -f bmh.yaml
```

出力例

```
secret/worker1-bmc created
baremetalhost.metal3.io/worker1 created
```

3. すべての証明書署名要求 (CSR) を承認します。

   a. 次のコマンドを実行して、ホストのプロビジョニング状態が **provisioned** であることを確認します。

```
$ oc get bmh -A
```

出力例

```
NAMESPACE               NAME        STATE                  CONSUMER   ONLINE
ERROR   AGE
openshift-machine-api   controller1   externally provisioned             true
5m25s
```

```
openshift-machine-api   worker1       provisioned                 true            4m45s
```

b. 次のコマンドを実行して、保留中の CSR のリストを取得します。

```
$ oc get csr
```

出力例

```
NAME        AGE   SIGNERNAME                                    REQUESTOR
REQUESTEDDURATION CONDITION
csr-gfm9f   33s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-o
perator:node-bootstrapper   <none>              Pending
```

c. 次のコマンドを実行して、CSR を承認します。

```
$ oc adm certificate approve <csr_name>
```

出力例

```
certificatesigningrequest.certificates.k8s.io/<csr_name> approved
```

検証

- 次のコマンドを実行して、ノードの準備ができていることを確認します。

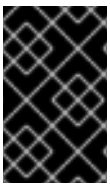
```
$ oc get nodes
```

出力例

```
NAME        STATUS   ROLES          AGE     VERSION
app1        Ready    worker         47s     v1.24.0+dc5a2fd
controller1 Ready    master,worker   2d22h   v1.24.0+dc5a2fd
```

15.5.4. オプション: BMO を使用して、ユーザーがプロビジョニングしたクラスターで既存のホストを管理する

オプションで、Bare Metal Operator (BMO) を使用して、既存のホストの **BareMetalHost** オブジェクトを作成すると、ユーザーがプロビジョニングしたクラスターで既存のベアメタルコントローラーホストを管理できます。ユーザーがプロビジョニングした既存のホストを管理する必要はありません。ただし、それらをインベントリー目的で外部プロビジョニングされたホストとして登録することはできます。



重要

BMO を使用して、既存のホストを管理するには、**BareMetalHost** カスタムリソースの **spec.externallyProvisioned** 仕様を **true** に設定して、BMO がホストを再プロビジョニングしないようにする必要があります。

前提条件

- ユーザーがプロビジョニングしたベアメタルクラスターを作成しました。
- ホストへのベースボード管理コントローラー (BMC) アクセス権限がある。
- **Provisioning** CR を作成して、クラスターにプロビジョニングサービスをデプロイしました。

手順

1. **Secret** CR と **BareMetalHost** CR を作成します。
   a. 次の YAML を **controller.yaml** ファイルに保存します。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: controller1-bmc
  namespace: openshift-machine-api
type: Opaque
data:
  username: <base64_of_uid>
  password: <base64_of_pwd>
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: controller1
  namespace: openshift-machine-api
spec:
  bmc:
    address: <protocol>://<bmc_url> (1)
    credentialsName: "controller1-bmc"
  bootMACAddress: <nic1_mac_address>
  customDeploy:
    method: install_coreos
  externallyProvisioned: true (2)
  online: true
  userData:
    name: controller-user-data-managed
    namespace: openshift-machine-api
```

(1) 仮想メディアネットワークの起動をサポートするベアメタルホストドライバー (**redfish-virtualmedia** や **idrac-virtualmedia** など) のみを使用できます。

(2) BMO がベアメタルコントローラーホストを再プロビジョニングしないようにするには、値を true に設定する必要があります。

2. 次のコマンドを実行して、ベアメタルホストオブジェクトを作成します。

```
$ oc create -f controller.yaml
```

出力例

```
secret/controller1-bmc created
baremetalhost.metal3.io/controller1 created
```

検証

- 次のコマンドを実行して、BMO がベアメタルホストオブジェクトを作成したことを確認します。

```
$ oc get bmh -A
```

出力例

```
NAMESPACE              NAME         STATE                  CONSUMER   ONLINE   ERROR
AGE
openshift-machine-api  controller1  externally provisioned             true
13s
```

15.5.5. BMO を使用して、ユーザーがプロビジョニングしたクラスターからホストを削除する

Bare Metal Operator (BMO) を使用して、ユーザーがプロビジョニングしたクラスターからベアメタルホストを削除できます。

前提条件

- ユーザーがプロビジョニングしたベアメタルクラスターを作成しました。
- ホストへのベースボード管理コントローラー (BMC) アクセス権限がある。
- **Provisioning** CR を作成して、クラスターにプロビジョニングサービスをデプロイしました。

手順

1. 次のコマンドを実行して、ホストを封鎖およびドレインします。

```
$ oc adm drain app1 --force --ignore-daemonsets=true
```

出力例

```
node/app1 cordoned
WARNING: ignoring DaemonSet-managed Pods: openshift-cluster-node-tuning-
operator/tuned-tvthg, openshift-dns/dns-
default-9q6rz, openshift-dns/node-resolver-zvt42, openshift-image-registry/node-ca-mzxth,
openshift-ingress-cana
ry/ingress-canary-qq5lf, openshift-machine-config-operator/machine-config-daemon-v79dm,
openshift-monitoring/nod
e-exporter-2vn59, openshift-multus/multus-additional-cni-plugins-wssvj, openshift-
multus/multus-fn8tg, openshift
-multus/network-metrics-daemon-5qv55, openshift-network-diagnostics/network-check-
target-jqxn2, openshift-ovn-ku
bernetes/ovnkube-node-rsvqg
evicting pod openshift-operator-lifecycle-manager/collect-profiles-27766965-258vp
evicting pod openshift-operator-lifecycle-manager/collect-profiles-27766950-kg5mk
evicting pod openshift-operator-lifecycle-manager/collect-profiles-27766935-stf4s
```

```
pod/collect-profiles-27766965-258vp evicted
pod/collect-profiles-27766950-kg5mk evicted
pod/collect-profiles-27766935-stf4s evicted
node/app1 drained
```

2. **BareMetalHost** CR から **customDeploy** 仕様を削除します。

   a. 次のコマンドを実行して、ホストの **BareMetalHost** CR を編集します。

   ```
   $ oc edit bmh -n openshift-machine-api <host_name>
   ```

   b. **spec.customDeploy** および **spec.customDeploy.method** の行を削除します。

   ```
   ...
     customDeploy:
       method: install_coreos
   ```

   c. 次のコマンドを実行して、ホストのプロビジョニング状態が **deprovisioning** に変わることを確認します。

   ```
   $ oc get bmh -A
   ```

   出力例

   ```
   NAMESPACE               NAME        STATE                  CONSUMER   ONLINE
   ERROR   AGE
   openshift-machine-api   controller1   externally provisioned           true           58m
   openshift-machine-api   worker1     deprovisioning                 true           57m
   ```

3. 次のコマンドを実行して、ノードを削除します。

   ```
   $ oc delete node <node_name>
   ```

検証

- 次のコマンドを実行して、ノードが削除されたことを確認します。

  ```
  $ oc get nodes
  ```

  出力例

  ```
  NAME          STATUS   ROLES          AGE     VERSION
  controller1   Ready    master,worker   2d23h   v1.24.0+dc5a2fd
  ```

15.6. ベアメタルのインストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

15.6.1. ベアメタルで利用可能なインストール設定パラメーター

以下の表に、インストールプロセスの一部として設定できる必須、オプション、およびベアメタル固有のインストール設定パラメーターを示します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

15.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表15.42 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。

パラメーター	説明	値
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

15.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

- Red Hat OpenShift Networking OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 と IPv6 の両方のアドレスファミリーがサポートされます。

両方の IP アドレスファミリーを使用するようにクラスターを設定する場合は、次の要件を確認してください。

- どちらの IP ファミリーも、デフォルトゲートウェイに同じネットワークインターフェイスを使用する必要があります。
- 両方の IP ファミリーにデフォルトゲートウェイが必要です。
- すべてのネットワーク設定パラメーターに対して、IPv4 アドレスと IPv6 アドレスを同じ順序で指定する必要があります。たとえば、以下の設定では、IPv4 アドレスは IPv6 アドレスの前に記載されます。

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
```

```
- cidr: fd00:10:128::/56
  hostPrefix: 64
serviceNetwork:
- 172.30.0.0/16
- fd00:172:16::/112
```

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表15.43 ネットワークパラメーター

パラメーター	説明	値
`networking:`	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
`networking:` `  networkType:`	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
`networking:` `  clusterNetwork:`	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 `networking:` `  clusterNetwork:` `  - cidr: 10.128.0.0/14` `  hostPrefix: 23` `  - cidr: fd01::/48` `  hostPrefix: 64`
`networking:` `  clusterNetwork:` `  cidr:`	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 OVN-Kubernetes ネットワークプラグインを使用する場合は、IPv4 および IPv6 ネットワークを指定できます。	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。IPv6 ブロックの接頭辞長は **0** から **128** までです。例: **10.128.0.0/14** または **fd01::/48**

パラメーター	説明	値
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 IPv4 ネットワークの場合、デフォルト値は **23** です。IPv6 ネットワークの場合、デフォルト値は **64** です。デフォルト値は、IPv6 の最小値でもあります。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。 OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 および IPv6 アドレスファミリーの両方に IP アドレスブロックを指定できます。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16 - fd02::/112
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** または **fd00::/48** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

15.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

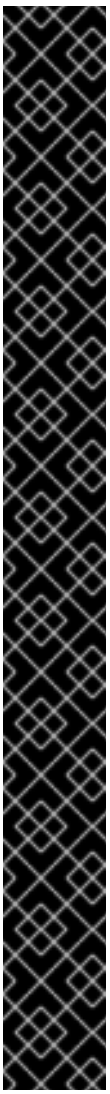
表15.44 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** と **arm64** です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。

パラメーター	説明	値
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** と **arm64** です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。 **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。

パラメーター	説明	値
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]

パラメーター	説明	値
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

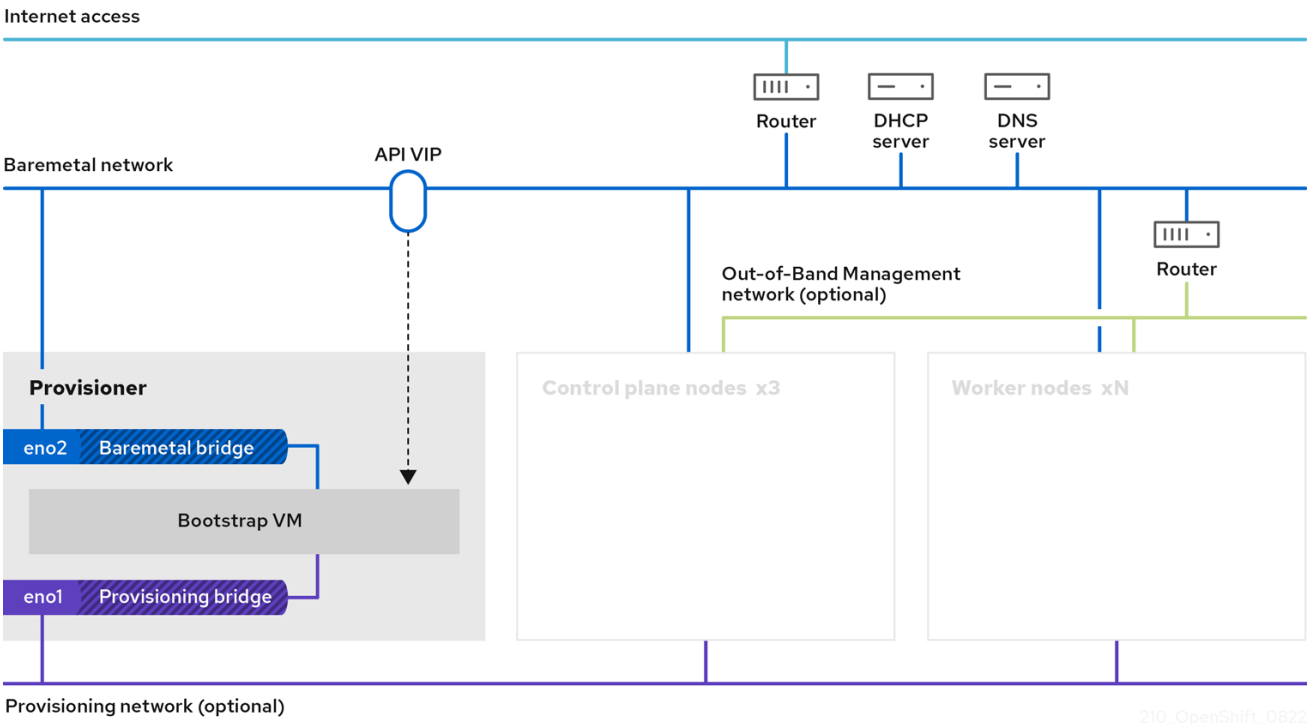
パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。 **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

第16章 インストーラーでプロビジョニングされるクラスターのベアメタルへのデプロイ

16.1. 概要

ベアメタルノードへのインストーラーによってプロビジョニングされたインストールは、OpenShift Container Platform クラスターが実行されるインフラストラクチャーをデプロイおよび設定します。このガイドでは、インストーラーによってプロビジョニングされたベアメタルのインストールを正常に行うための方法論を提供します。次の図は、デプロイメントのフェーズ 1 におけるインストール環境を示しています。



インストールの場合、前の図の主要な要素は次のとおりです。

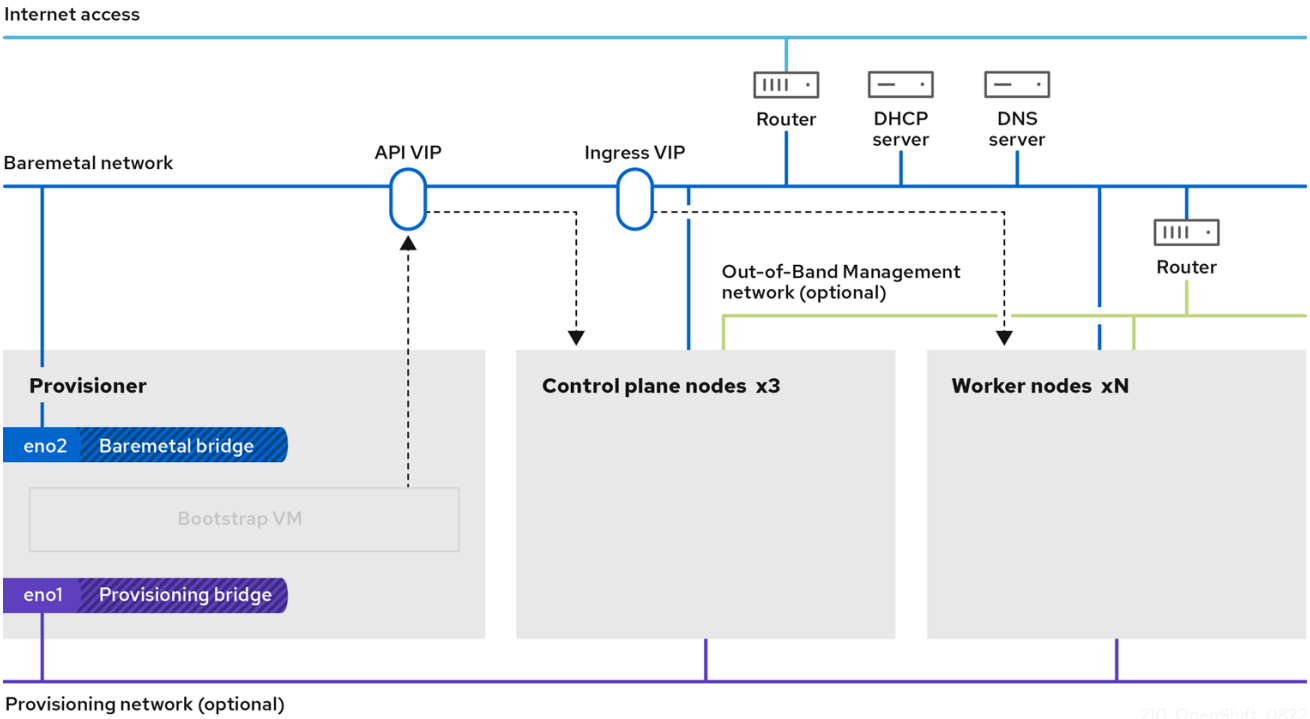
- **プロビジョナー**: インストールプログラムを実行し、新しい OpenShift Container Platform クラスターのコントロールプレーンをデプロイするブートストラップ VM をホストする物理マシン。
- **Bootstrap VM**: OpenShift Container Platform クラスターのデプロイプロセスで使用される仮想マシン。
- **ネットワークブリッジ**: ブートストラップ VM は、ネットワークブリッジ **eno1** および **eno2** を介して、ベアメタルネットワークとプロビジョニングネットワーク (存在する場合) に接続します。
- **API VIP**: API 仮想 IP アドレス (VIP) は、コントロールプレーンノード全体で API サーバーのフェイルオーバーを提供するために使用されます。API VIP はまずブートストラップ仮想マシンにあります。スクリプトは、サービスを起動する前に **keepalived.conf** 設定ファイルを生成します。VIP は、ブートストラッププロセスが完了し、ブートストラップ仮想マシンが停止すると、コントロールプレーンノードの 1 つに移動します。

デプロイメントのフェーズ 2 では、プロビジョナーがブートストラップ VM を自動的に破棄し、仮想 IP アドレス (VIP) を適切なノードに移動します。

keepalived.conf ファイルは、コントロールプレーンマシンにブートストラップ VM よりも低い仮想ルーター冗長プロトコル (VRRP) の優先順位を設定します。これにより、API VIP がブートストラップ VM からコントロールプレーンに移動する前に、コントロールプレーンマシン上の API が完全に機能することが保証されます。API VIP がコントロールプレーンノードの 1 つに移動すると、外部クライアントから API VIP ルートに送信されたトラフィックが、そのコントロールプレーンノードで実行している **haproxy** ロードバランサーに移動します。**haproxy** のこのインスタンスは、コントロールプレーンノード全体で API VIP トラフィックを分散します。

Ingress 仮想 IP はコンピュートノードに移動します。**keepalived** インスタンスは Ingress VIP も管理します。

次の図は、デプロイメントのフェーズ 2 を示しています:



これ以降、プロビジョナーが使用するノードを削除または再利用できます。ここから、追加のプロビジョニングタスクはすべてコントロールプレーンによって実行されます。

注記

installer-provisioned infrastructure によるインストールの場合、ポート 53 が CoreDNS によってノードレベルで公開され、他のルーティング可能なネットワークからアクセスできるようになります。

関連情報

- DNS 転送の使用

重要

プロビジョニングネットワークは任意ですが、PXE ブートには必要です。プロビジョニングネットワークなしでデプロイする場合は、**redfish-virtualmedia** または **idrac-virtualmedia** などの仮想メディアベースボード管理コントローラー (BMC) アドレス指定オプションを使用する必要があります。

16.2. 前提条件

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールには、以下が必要です。

1. 1 つの Red Hat Enterprise Linux (RHEL) 9.x がインストールされているプロビジョナーノード。プロビジョナーは、インストール後に削除できます。
2. 3 つのコントロールプレーンノード
3. ベースボード管理コントローラー (BMC) の各ノードへのアクセス。
4. 1 つ以上のネットワーク:
    a. 1 つの必須 のルーティング可能なネットワーク
    b. 1 つのオプションのプロビジョニングネットワーク
    c. 1 つのオプションの管理ネットワーク

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールを開始する前に、ハードウェア環境が以下の要件を満たしていることを確認してください。

16.2.1. ノードの要件

インストーラーでプロビジョニングされるインストールには、ハードウェアノードの各種の要件があります。

- **CPU architecture:** すべてのノードは **x86_64** または **aarch64** CPU アーキテクチャーを使用する必要があります。
- **同様のノード:** Red Hat では、ノードにロールごとに同じ設定を指定することを推奨しています。つまり Red Hat では、同じ CPU、メモリー、ストレージ設定の同じブランドおよびモデルのノードを使用することを推奨しています。
- **ベースボード管理コントローラー: provisioner** ノードは、各 OpenShift Container Platform クラスターノードのベースボード管理コントローラー (BMC) にアクセスできる必要があります。IPMI、RedFish、または独自のプロトコルを使用できます。
- **最新の生成:** ノードは最新の生成されたノードである必要があります。インストーラーでプロビジョニングされるインストールは、ノード間で互換性を確保する必要のある BMC プロトコルに依存します。また、RHEL 9 には RAID コントローラーの最新のドライバーが同梱されています。ノードが、**provisioner** ノードの RHEL 9.x と、コントロールプレーンおよびワーカーノードの RHCOS 9.x をサポートするために必要な新しいバージョンのノードでであることを確認します。
- **レジストリーノード:** (オプション) 非接続のミラーリングされていないレジストリーを設定する場合、レジストリーは独自のノードに置くことが推奨されます。
- **プロビジョナーノード:** インストーラーでプロビジョニングされるインストールには 1 つの **provisioner** ノードが必要です。
- **コントロールプレーン:** インストーラーでプロビジョニングされるインストールには、高可用性を実現するために 3 つのコントロールプレーンノードが必要です。3 つのコントロールプレーンノードのみで OpenShift Container Platform クラスターをデプロイして、コントロールプ

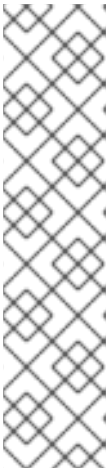
レーンノードをワーカーノードとしてスケジュール可能にすることができます。小規模なクラスターでは、開発、実稼働およびテスト時の管理者および開発者に対するリソース効率が高くなります。

- **ワーカーノード:** 必須ではありませんが、一般的な実稼働クラスターには複数のワーカーノードがあります。

重要

ワーカーノードが1つしかないクラスターは、パフォーマンスが低下した状態のルーターおよびIngressトラフィックでデプロイされるので、デプロイしないでください。

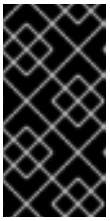
- **ネットワークインターフェイス:** 各ノードでは、ルーティング可能な **baremetal** ネットワークに1つ以上のネットワークインターフェイスが必要です。デプロイメントに **provisioning** ネットワークを使用する場合、各ノードに **provisioning** ネットワーク用に1つのネットワークインターフェイスが必要になります。**provisioning** ネットワークの使用はデフォルト設定です。



注記

同じサブネット上の1つのネットワークカード (NIC) のみが、ゲートウェイ経由でトラフィックをルーティングできます。デフォルトでは、アドレス解決プロトコル (ARP) は番号が最も小さい NIC を使用します。ネットワーク負荷分散が期待どおりに機能するようにするには、同じサブネット内の各ノードに1つの NIC を使用します。同じサブネット内のノードに複数の NIC を使用する場合は、単一のボンディングまたはチームインターフェイスを使用します。次に、エイリアス IP アドレスの形式で他の IP アドレスをそのインターフェイスに追加します。ネットワークインターフェイスレベルでフォールトトレランスまたは負荷分散が必要な場合は、ボンディングまたはチームインターフェイスでエイリアス IP アドレスを使用します。または、同じサブネットでセカンダリー NIC を無効にするか、IP アドレスがないことを確認できます。

- **Unified Extensible Firmware Interface (UEFI):**インストーラーでプロビジョニングされるインストールでは、**provisioning** ネットワークで IPv6 アドレスを使用する場合に、すべての OpenShift Container Platform ノードで UEFI ブートが必要になります。さらに、UEFI Device PXE Settings (UEFI デバイス PXE 設定) は **provisioning** ネットワーク NIC で IPv6 プロトコルを使用するように設定する必要がありますが、**provisioning** ネットワークを省略すると、この要件はなくなります。



重要

ISO イメージなどの仮想メディアからインストールを開始する場合は、古い UEFI ブートテーブルエントリーをすべて削除します。ブートテーブルに、ファームウェアによって提供される一般的なエントリーではないエントリーが含まれている場合、インストールは失敗する可能性があります。

- **Secure Boot:** 多くの実稼働シナリオでは、UEFI ファームウェアドライバー、EFI アプリケーション、オペレーティングシステムなど、信頼できるソフトウェアのみを使用してノードが起動することを確認するために、Secure Boot が有効にされているノードが必要です。手動またはマネージドの Secure Boot を使用してデプロイすることができます。

1. **手動:** 手動で Secure Boot を使用して OpenShift Container Platform クラスターをデプロイするには、UEFI ブートモードおよび Secure Boot を各コントロールプレーンノードおよび各ワーカーノードで有効にする必要があります。Red Hat は、インストーラーでプロビ

ジョニングされるインストールで Redfish 仮想メディアを使用している場合にのみ、手動で有効にした UEFI および Secure Boot で、Secure Boot をサポートします。詳細は、ノードの設定セクションの手動での Secure Boot のノードの設定を参照してください。

2. **マネージド:** マネージド Secure Boot で OpenShift Container Platform クラスターをデプロイするには、**install-config.yaml** ファイルで **bootMode** の値を **UEFISecureBoot** に設定する必要があります。Red Hat は、第 10 世代 HPE ハードウェアおよび 13 世代 Dell ハードウェア (ファームウェアバージョン **2.75.75.75** 以上を実行) でマネージド Secure Boot を使用したインストーラーでプロビジョニングされるインストールのみをサポートします。マネージド Secure Boot を使用したデプロイには、Redfish 仮想メディアは必要ありません。詳細は、OpenShift インストールの環境のセットアップセクションのマネージド Secure Boot の設定を参照してください。

注記

Red Hat は、自己生成したキーを使用する Secure Boot をサポートしません。

16.2.2. OpenShift 仮想化のためのベアメタルクラスターの計画

OpenShift 仮想化を使用する場合は、ベアメタルクラスターをインストールする前に、いくつかの要件を認識することが重要です。

- ライブマイグレーション機能を使用する場合は、**クラスターのインストール時に** 複数のワーカーノードが必要です。これは、ライブマイグレーションではクラスターレベルの高可用性 (HA) フラグを true に設定する必要があるためです。HA フラグは、クラスターのインストール時に設定され、後で変更することはできません。クラスターのインストール時に定義されたワーカーノードが 2 つ未満の場合、クラスターの存続期間中、HA フラグは false に設定されます。

注記

単一ノードのクラスターに OpenShift Virtualization をインストールできますが、単一ノードの OpenShift は高可用性をサポートしていません。

- ライブマイグレーションには共有ストレージが必要です。OpenShift Virtualization のストレージは、ReadWriteMany (RWX) アクセスモードをサポートし、使用する必要があります。
- Single Root I/O Virtualization (SR-IOV) を使用する予定の場合は、ネットワークインターフェイスコントローラー (NIC) が OpenShift Container Platform でサポートされていることを確認してください。

関連情報

- OpenShift Virtualization のクラスターの準備
- Single Root I/O Virtualization (SR-IOV) ハードウェアネットワークについて
- 仮想マシンの SR-IOV ネットワークへの接続

16.2.3. 仮想メディアを使用したインストールのファームウェア要件

インストーラーでプロビジョニングされる OpenShift Container Platform クラスターのインストールプログラムは、Redfish 仮想メディアとのハードウェアおよびファームウェアの互換性を検証します。

ノードファームウェアに互換性がない場合、インストールプログラムはノードへのインストールを開始しません。以下の表は、Redfish 仮想メディアを使用してデプロイされるインストーラーでプロビジョニングされる OpenShift Container Platform クラスターで機能するようにテストおよび検証された最小ファームウェアバージョンの一覧です。

注記

Red Hat は、ファームウェア、ハードウェア、またはその他のサードパーティーコンポーネントの組み合わせをすべてテストしません。サードパーティーサポートの詳細は、Red Hat サードパーティーサポートポリシー を参照してください。ファームウェアの更新については、ノードのハードウェアドキュメントを参照するか、ハードウェアベンダーにお問い合わせください。

表16.1 HP ハードウェアと Redfish 仮想メディアのファームウェアの互換性

モデル	管理	ファームウェアのバージョン
第 10 世代	iLO5	2.63 以降

表16.2 Redfish 仮想メディアを使用した Dell ハードウェアのファームウェアの互換性

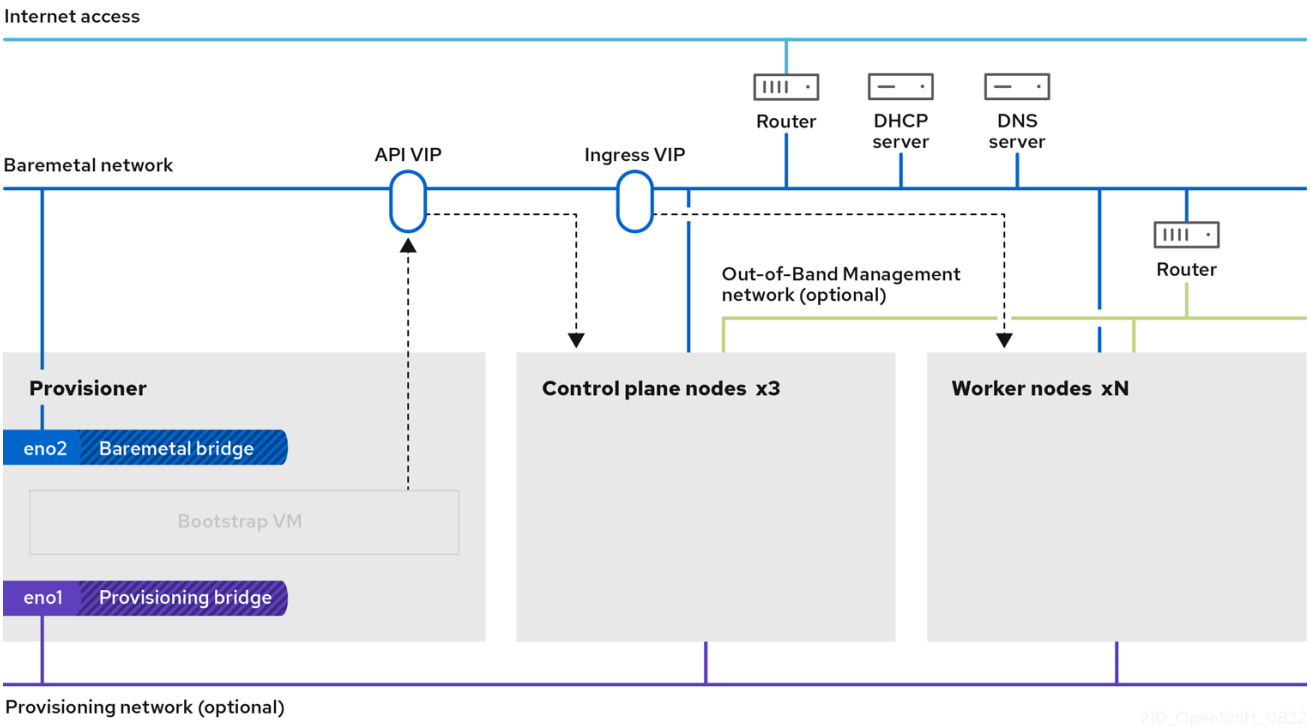
モデル	管理	ファームウェアのバージョン
第 15 世代	iDRAC 9	v6.10.30.00 および v7.00.60.00
第 14 世代	iDRAC 9	v6.10.30.00
第 13 世代	iDRAC 8	v2.75.75.75 以降

関連情報

BMC を使用して、新しいベアメタルホストを検出できない

16.2.4. ネットワーク要件

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールには、複数のネットワーク要件があります。まず、インストーラーでプロビジョニングされるインストールでは、各ベアメタルノードにオペレーティングシステムをプロビジョニングするためのルーティング不可能な **provisioning** ネットワークをオプションで使用します。次に、インストーラーでプロビジョニングされるインストールでは、ルーティング可能な **baremetal** ネットワークを使用します。



16.2.4.1. 必須ポートが開いていることを確認する

クラスター間で特定のノードが開いてなければ、installer-provisioned によるインストールは正常に完了しません。ファーエッジワーカーノードに別のサブネットを使用する場合などの特定の状況では、これらのサブネット内のノードが、次の必須ポートで他のサブネット内のノードと通信できることを確認する必要があります。

表16.3 必須ポート

ポート	説明
67、68	プロビジョニングネットワークを使用する場合、クラスターノードは、ポート **67** および **68** を使用して、プロビジョニングネットワークインターフェイス経由で **dnsmasq** DHCP サーバーにアクセスします。
69	プロビジョニングネットワークを使用する場合、クラスターノードはプロビジョニングネットワークインターフェイスを使用してポート **69** で TFTP サーバーと通信します。TFTP サーバーはブートストラップ仮想マシン上で実行されます。ブートストラップ仮想マシンはプロビジョナーノード上で実行されます。

ポート	説明
80	イメージキャッシュオプションを使用しない場合、または仮想メディアを使用する場合、プロビジョナーノードからクラスターノードに Red Hat Enterprise Linux CoreOS (RHCOS) イメージをストリーミングするには、プロビジョナーノードのポート **80** が **baremetal** マシンのネットワークインターフェイス上で開いている必要があります。
123	クラスターノードは、**baremetal** マシンネットワークを使用して、ポート **123** の NTP サーバーにアクセスする必要があります。
5050	Ironic Inspector API は、コントロールプレーンノード上で実行され、ポート **5050** でリッスンします。Inspector API は、ベアメタルノードのハードウェア特性に関する情報を収集するハードウェアイントロスペクションを担当します。
5051	ポート **5050** はポート **5051** をプロキシーとして使用します。
6180	仮想メディアを使用して TLS を使用せずにデプロイする場合、ワーカーノードのベースボード管理コントローラー (BMC) が RHCOS イメージにアクセスできるように、プロビジョナーノードとコントロールプレーンノードの ポート **6180** は、**baremetal** マシンのネットワークインターフェイスで開いている必要があります。OpenShift Container Platform 4.13 以降、デフォルトの HTTP ポートは **6180** です。
6183	仮想メディアを使用して TLS でデプロイする場合、ワーカーノードの BMC が RHCOS イメージにアクセスできるように、プロビジョナーノードとコントロールプレーンノードの ポート **6183** は、**baremetal** マシンのネットワークインターフェイスで開いている必要があります。
6385	Ironic API サーバーは、最初はブートストラップ仮想マシン上で実行され、その後はコントロールプレーンノード上で実行され、ポート **6385** でリッスンします。Ironic API を使用すると、クライアントは Ironic と対話して、新規ノードの登録、電源状態の管理、イメージのデプロイメント、ハードウェアのクリーニングなどの操作を含む、ベアメタルノードのプロビジョニングと管理を実行できます。
6388	ポート **6385** はポート **6388** をプロキシーとして使用します。

ポート	説明
8080	TLS なしでイメージキャッシュを使用する場合、ポート **8080** がプロビジョナーノードで開いており、クラスターノードの BMC インターフェイスからアクセスできる必要があります。
8083	TLS ありでイメージキャッシュオプションを使用する場合、ポート **8083** がプロビジョナーノードで開いており、クラスターノードの BMC インターフェイスからアクセスできる必要があります。
9999	デフォルトでは、Ironic Python Agent (IPA) は、ポート **9999** で Ironic conductor サービスからの API 呼び出しを TCP リッスンします。このポートは、IPA が実行されているベアメタルノードと Ironic conductor サービス間の通信に使用されます。

16.2.4.2. ネットワーク MTU の増加

OpenShift Container Platform をデプロイする前に、ネットワークの最大伝送単位 (MTU) を 1500 以上に増やします。MTU が 1500 未満の場合、ノードの起動に使用される Ironic イメージが Ironic インスペクター Pod との通信に失敗し、検査が失敗する可能性があります。これが発生すると、インストールにノードを使用できないため、インストールが停止します。

16.2.4.3. NIC の設定

OpenShift Container Platform は、2 つのネットワークを使用してデプロイします。

- **provisioning**: **provisioning** ネットワークは、OpenShift Container Platform クラスターの一部である基礎となるオペレーティングシステムを各ノードにプロビジョニングするために使用されるオプションのルーティング不可能なネットワークです。各クラスターノードの **provisioning** ネットワークのネットワークインターフェイスには、BIOS または UEFI が PXE ブートに設定されている必要があります。
  provisioningNetworkInterface 設定は、コントロールプレーンノード上の **provisioning** ネットワークの NIC 名を指定します。これは、コントロールプレーンノードと同じでなければなりません。**bootMACAddress** 設定は、**provisioning** ネットワーク用に各ノードで特定の NIC を指定する手段を提供します。

  provisioning ネットワークは任意ですが、PXE ブートには必要です。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。

- **baremetal**: **baremetal** ネットワークはルーティング可能なネットワークです。NIC が **provisioning** ネットワークを使用するように設定されていない場合には、**baremetal** ネットワークとのインターフェイスには任意の NIC を使用することができます。

重要

VLAN を使用する場合、それぞれの NIC は、適切なネットワークに対応する別個の VLAN 上にある必要があります。

16.2.4.4. DNS 要件

クライアントは、**baremetal** ネットワークで OpenShift Container Platform クラスターにアクセスします。ネットワーク管理者は、正規名の拡張がクラスター名であるサブドメインまたはサブゾーンを設定する必要があります。

```
<cluster_name>.<base_domain>
```

以下に例を示します。

```
test-cluster.example.com
```

OpenShift Container Platform には、クラスターメンバーシップ情報を使用して A/AAAA レコードを生成する機能が含まれます。これにより、ノード名が IP アドレスに解決されます。ノードが API に登録されると、クラスターは CoreDNS-mDNS を使用せずにこれらのノード情報を分散できます。これにより、マルチキャスト DNS に関連付けられたネットワークトラフィックがなくなります。

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード Ingress API

A/AAAA レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。Red Hat Enterprise Linux Core OS(RHCOS) は、逆引きレコードまたは DHCP を使用して、すべてのノードのホスト名を設定します。

インストーラーがプロビジョニングしたインストールには、クラスターメンバーシップ情報を使用して A/AAAA レコードを生成する機能が含まれています。これにより、ノード名が IP アドレスに解決されます。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表16.4 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	A/AAAA レコードと PTR レコード。API ロードバランサーを識別します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	ワイルドカード A/AAAA レコードは、アプリケーションの Ingress ロードバランサーを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するノードをターゲットにします。Ingress Controller Pod は、デフォルトでワーカーノードで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。

ヒント

digコマンドを使用して、DNS 解決を確認できます。

16.2.4.5. Dynamic Host Configuration Protocol (DHCP) の要件

デフォルトでは、インストーラーでプロビジョニングされるインストールは、**provisioning** ネットワーク用に DHCP を有効にして **ironic-dnsmasq** をデプロイします。**provisioningNetwork** 設定が、デフォルト値の **managed** に設定されている場合、**provisioning** ネットワーク上で他の DHCP サーバーを実行することはできません。**provisioning** ネットワーク上で DHCP サーバーを実行している場合は、**install-config.yaml** ファイルで **provisioningNetwork** 設定を **unmanaged** に設定する必要があります。

ネットワーク管理者は、外部 DHCP サーバー上の **baremetal** ネットワーク用に、OpenShift Container Platform クラスター内の各ノードの IP アドレスを予約する必要があります。

16.2.4.6. DHCP サーバーを使用するノードの IP アドレスの確保

baremetal ネットワークの場合、ネットワーク管理者は以下を含む多数の IP アドレスを予約する必要があります。

1. 2 つの一意の仮想 IP アドレス。
    - API エンドポイントの 1 つの仮想 IP アドレス。
    - ワイルドカード Ingress エンドポイントの 1 つの仮想 IP アドレス
2. プロビジョナーノードの 1 つの IP アドレス
3. コントロールプレーンノードごとに 1 つの IP アドレス。
4. 各ワーカーノードの 1 つの IP アドレス (適用可能な場合)

IP アドレスの予約し、それらを静的 IP アドレスにする

一部の管理者は、各ノードの IP アドレスが DHCP サーバーがない状態で一定になるように静的 IP アドレスの使用を選択します。NMState を使用して静的 IP アドレスを設定する場合は、「OpenShift インストールの環境のセットアップ」セクションの「ホストネットワークインターフェイスの設定 (オプション)」を参照してください。

外部ロードバランサーとコントロールプレーンノード間のネットワーク

外部の負荷分散サービスとコントロールプレーンノードは同じ L2 ネットワークで実行する必要があります。また、VLAN を使用して負荷分散サービスとコントロールプレーンノード間のトラフィックをルーティングする際に同じ VLAN で実行する必要があります。

重要

ストレージインターフェイスには、DHCP 予約または静的 IP が必要です。

以下の表は、完全修飾ドメイン名の具体例を示しています。API および Nameserver アドレスは、正式名の拡張子で始まります。コントロールプレーンおよびワーカーノードのホスト名は例であるため、任意のホストの命名規則を使用することができます。

使用法	ホスト名	IP
API	**api.<cluster_name>.<base_domain>**	**<ip>**
Ingress LB (アプリケーション)	***.apps.<cluster_name>.<base_domain>**	**<ip>**
プロビジョナーノード	**provisioner.<cluster_name>.<base_domain>**	**<ip>**
Control-plane-0	**openshift-control-plane-0.<cluster_name>.<base_domain>**	**<ip>**
Control-plane-1	**openshift-control-plane-1.<cluster_name>-.<base_domain>**	**<ip>**
Control-plane-2	**openshift-control-plane-2.<cluster_name>.<base_domain>**	**<ip>**
Worker-0	**openshift-worker-0.<cluster_name>.<base_domain>**	**<ip>**
Worker-1	**openshift-worker-1.<cluster_name>.<base_domain>**	**<ip>**
Worker-n	**openshift-worker-n.<cluster_name>.<base_domain>**	**<ip>**

注記

DHCP 予約を作成しない場合には、インストーラーは、Kubernetes API ノード、プロビジョナーノード、コントロールプレーンノード、およびワーカーノードのホスト名を設定するために逆引き DNS 解決を必要とします。

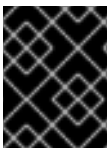
16.2.4.7. プロビジョナーノードの要件

インストール設定でプロビジョナーノードの MAC アドレスを指定する必要があります。通常、**bootMacAddress** 仕様は PXE ネットワークブートに関連付けられています。ただし、Ironic プロビジョニングサービスでは、クラスターの検査中またはクラスター内のノードの再デプロイ中にノードを識別するために、**bootMacAddress** 仕様も必要です。

プロビジョナーノードには、ネットワークの起動、DHCP と DNS の解決、およびローカルネットワーク通信のためのレイヤー 2 接続が必要です。プロビジョナーノードには、仮想メディアの起動にレイヤー 3 接続が必要です。

16.2.4.8. ネットワークタイムプロトコル (NTP)

クラスター内の各 OpenShift Container Platform ノードは NTP サーバーにアクセスできる必要があります。OpenShift Container Platform ノードは NTP を使用してクロックを同期します。たとえば、クラスターノードは、検証を必要とする SSL 証明書を使用します。これは、ノード間の日付と時刻が同期していない場合に失敗する可能性があります。



重要

各クラスターノードの BIOS 設定で一貫性のあるクロックの日付と時刻の形式を定義してください。そうしないと、インストールが失敗する可能性があります。

切断されたクラスター上で NTP サーバーとして機能するようにコントロールプレーンノードを再設定し、コントロールプレーンノードから時間を取得するようにワーカーノードを再設定することができます。

16.2.4.9. 帯域外管理 IP アドレスのポートアクセス

帯域外管理 IP アドレスは、ノードとは別のネットワーク上にあります。インストール中に帯域外管理がプロビジョナーノードと確実に通信できるようにするには、帯域外管理 IP アドレスに、プロビジョナーノードおよび OpenShift Container Platform コントロールプレーンノード上のポート **6180** へのアクセスを許可する必要があります。TLS ポート **6183** は、たとえば Redfish などを使用する仮想メディアのインストールに必要です。

関連情報

- DNS 転送の使用

16.2.5. ノードの設定

provisioning ネットワークを使用する場合のノードの設定
クラスター内の各ノードには、適切なインストールを行うために以下の設定が必要です。

警告

ノード間で一致していないと、インストールに失敗します。

クラスターノードには 3 つ以上の NIC を追加できますが、インストールプロセスでは最初の 2 つの NIC のみに焦点が当てられます。以下の表では、NIC1 は OpenShift Container Platform クラスターのインストールにのみ使用されるルーティング不可能なネットワーク (**provisioning**) です。

NIC	ネットワーク	VLAN
NIC1	**provisioning**	**<provisioning_vlan>**
NIC2	**baremetal**	**<baremetal_vlan>**

プロビジョナーノードでの Red Hat Enterprise Linux (RHEL) 9.x インストールプロセスは異なる可能性があります。ローカルの Satellite サーバー、PXE サーバー、PXE 対応の NIC2 を使用して Red Hat Enterprise Linux (RHEL) 9.x をインストールする場合は、以下のようになります。

PXE	ブート順序
NIC1 PXE 対応の **provisioning** ネットワーク	1
NIC2 **baremetal** ネットワークPXE 対応はオプションです。	2

注記

他のすべての NIC で PXE が無効になっていることを確認します。

コントロールプレーンおよびワーカーノードを以下のように設定します。

PXE	ブート順序
NIC1 PXE 対応 (プロビジョニングネットワーク)	1

provisioning ネットワークを使用しないノードの設定
インストールプロセスには、1 つの NIC が必要です。

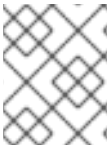
NIC	ネットワーク	VLAN
NICx	**baremetal**	**<baremetal_vlan>**

NICx は、OpenShift Container Platform クラスターのインストールに使用されるルーティング可能なネットワーク (**baremetal**) であり、インターネットにルーティング可能です。

重要

provisioning ネットワークは任意ですが、PXE ブートには必要です。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。

手動での Secure Boot のノードの設定
Secure Boot は、ノードが UEFI ファームウェアドライバー、EFI アプリケーション、オペレーティングシステムなどの信頼できるソフトウェアのみを使用していることを確認できない場合は、ノードの起動を阻止します。



注記

Red Hat は、RedFish 仮想メディアを使用してデプロイする場合にのみ、手動で設定された Secure Boot をサポートします。

Secure Boot を手動で有効にするには、ノードのハードウェアガイドを参照し、以下を実行してください。

手順

1. ノードを起動し、BIOS メニューを入力します。
2. ノードのブートモードを **UEFI Enabled** に設定します。
3. Secure Boot を有効にします。

重要

Red Hat は、自己生成したキーを使用する Secure Boot をサポートしません。

16.2.6. アウトオブバンド管理 (Out-of-band Management: 帯域外管理)

ノードには通常、ベースボード管理コントローラー (BMC) が使用する追加の NIC があります。これらの BMC は provisioner ノードからアクセスできる必要があります。

各ノードは、アウトバウンド管理でアクセスできるようにする必要があります。アウトバウンド管理ネットワークを使用する場合、 provisioner ノードには、OpenShift Container Platform の正常なインストールを実行するためにアウトバウンドネットワークへのアクセスが必要になります。

このアウトバウンド管理設定については、本書では扱いません。帯域外管理に個別の管理ネットワークを使用すると、パフォーマンスが向上し、セキュリティーが向上します。ただし、provisioning ネットワークまたはベアメタルネットワークの使用は有効なオプションになります。

注記

ブートストラップ仮想マシンには、最大 2 つのネットワークインターフェイスがあります。帯域外管理用に別の管理ネットワークを設定し、プロビジョニングネットワークを使用している場合、ブートストラップ仮想マシンでは、いずれかのネットワークインターフェイスを介して管理ネットワークへのルーティングアクセスが必要になります。このシナリオでは、ブートストラップ仮想マシンは 3 つのネットワークにアクセスできます。

- ベアメタルネットワーク
- プロビジョニングネットワーク
- 管理インターフェイスの 1 つを介してルーティングされる管理ネットワーク

16.2.7. インストールに必要なデータ

OpenShift Container Platform クラスターのインストール前に、すべてのクラスターノードから以下の情報を収集します。

- アウトバウンド管理 IP
  - 例
    - Dell (iDRAC) IP
    - HP (iLO) IP
    - Fujitsu (iRMC) IP

provisioning ネットワークを使用する場合

- NIC (**provisioning**) MAC アドレス
- NIC (**baremetal**) MAC アドレス

provisioning ネットワークを省略する場合

- NIC (**baremetal**) MAC アドレス

16.2.8. ノードの検証チェックリスト

provisioning ネットワークを使用する場合

❑ NIC1 VLAN が **provisioning** ネットワークについて設定されている。

❑ **provisioning** ネットワークの NIC1 は、プロビジョナー、コントロールプレーン、およびワーカーノードで PXE 対応として使用できる。

❑ NIC2 VLAN が **baremetal** ネットワークについて設定されている。

❑ PXE が他のすべての NIC で無効にされている。

❑ DNS は API および Ingress エンドポイントで設定されている。

❑ コントロールプレーンおよびワーカーノードが設定されている。

❑ すべてのノードがアウトオブバンド管理 (Out-of-band Management: 帯域外管理) でアクセス可能である。

❑ (オプション) 別個の管理ネットワークが作成されている。

❑ インストールに必要なデータ。

provisioning ネットワークを省略する場合

❑ NIC1 VLAN が **baremetal** ネットワークについて設定されている。

❑ DNS は API および Ingress エンドポイントで設定されている。

❑ コントロールプレーンおよびワーカーノードが設定されている。

❑ すべてのノードがアウトオブバンド管理 (Out-of-band Management: 帯域外管理) でアクセス可能である。

❑ (オプション) 別個の管理ネットワークが作成されている。

❑ インストールに必要なデータ。

16.3. OPENSHIFT インストールの環境のセットアップ

16.3.1. RHEL のプロビジョナーノードへのインストール

前提条件の設定が完了したら、次のステップとして RHEL 9.x をプロビジョナーノードにインストールします。インストーラーは、OpenShift Container Platform クラスターをインストールする間にプロビジョナーノードをオーケレーターとして使用します。本書の目的上、RHEL のプロビジョナーノードへのインストールは対象外です。ただし、オプションには、RHEL Satellite サーバー、PXE、またはインストールメディアの使用も含まれますが、これらに限定されません。

16.3.2. OpenShift Container Platform インストールのプロビジョナーノードの準備

環境を準備するには、以下の手順を実行します。

手順

1. **ssh** でプロビジョナーノードにログインします。
2. root 以外のユーザー (**kni**) を作成し、そのユーザーに **sudo** 権限を付与します。

```
# useradd kni
```

```
# passwd kni
```

```
# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
```

```
# chmod 0440 /etc/sudoers.d/kni
```

3. 新規ユーザーの **ssh** キーを作成します。

```
# su - kni -c "ssh-keygen -t ed25519 -f /home/kni/.ssh/id_rsa -N ""
```

4. プロビジョナーノードで新規ユーザーとしてログインします。

```
# su - kni
```

5. Red Hat Subscription Manager を使用してプロビジョナーノードを登録します。

```
$ sudo subscription-manager register --username=<user> --password=<pass> --auto-attach
```

```
$ sudo subscription-manager repos --enable=rhel-9-for-<architecture>-appstream-rpms --
enable=rhel-9-for-<architecture>-baseos-rpms
```

注記

Red Hat Subscription Manager についての詳細は、Using and Configuring Red Hat Subscription Manager を参照してください。

6. 以下のパッケージをインストールします。

```
$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```

7. ユーザーを変更して、新たに作成したユーザーに **libvirt** グループを追加します。

```
$ sudo usermod --append --groups libvirt <user>
```

8. **firewalld** を再起動して、**http** サービスを有効にします。

```
$ sudo systemctl start firewalld
```

```
$ sudo firewall-cmd --zone=public --add-service=http --permanent
```

```
$ sudo firewall-cmd --reload
```

9. **libvirtd** サービスを開始して、これを有効にします。

```
$ sudo systemctl enable libvirtd --now
```

10. **default** ストレージプールを作成して、これを起動します。

```
$ sudo virsh pool-define-as --name default --type dir --target /var/lib/libvirt/images
```

```
$ sudo virsh pool-start default
```

```
$ sudo virsh pool-autostart default
```

11. **pull-secret.txt** ファイルを作成します。

```
$ vim pull-secret.txt
```

Web ブラウザーで、Install OpenShift on Bare Metal with installer-provisioned infrastructure に移動します。**Copy pull secret** をクリックします。**pull-secret.txt** ファイルにコンテンツを貼り付け、そのコンテンツを **kni** ユーザーのホームディレクトリーに保存します。

16.3.3. NTP サーバーの同期を確認する

OpenShift Container Platform インストールプログラムは、**chrony** Network Time Protocol (NTP) サービスをクラスターノードにインストールします。インストールを完了するには、各ノードが NTP タイムサーバーにアクセスできる必要があります。**chrony** サービスを使用して、NTP サーバーの同期を確認できます。

切断されたクラスターの場合は、コントロールプレーンノード上で NTP サーバーを設定する必要があります。詳細は、**関連情報** セクションを参照してください。

前提条件

- ターゲットノードに **chrony** パッケージがインストールされました。

手順

1. **ssh** コマンドを使用してノードにログインします。
2. 次のコマンドを実行して、ノードで利用可能な NTP サーバーを表示します。

```
$ chronyc sources
```

出力例

```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^+ time.cloudflare.com           3  10   377   187   -209us[ -209us] +/-   32ms
^+ t1.time.ir2.yahoo.com         2  10   377   185  -4382us[-4382us] +/-   23ms
^+ time.cloudflare.com           3  10   377   198   -996us[-1220us] +/-   33ms
^* brenbox.westnet.ie            1  10   377   193  -9538us[-9761us] +/-   24ms
```

3. **ping** コマンドを使用して、ノードが NTP サーバーにアクセスできることを確認します。次に例を示します。

```
$ ping time.cloudflare.com
```

出力例

```
PING time.cloudflare.com (162.159.200.123) 56(84) bytes of data.
64 bytes from time.cloudflare.com (162.159.200.123): icmp_seq=1 ttl=54 time=32.3 ms
64 bytes from time.cloudflare.com (162.159.200.123): icmp_seq=2 ttl=54 time=30.9 ms
64 bytes from time.cloudflare.com (162.159.200.123): icmp_seq=3 ttl=54 time=36.7 ms
...
```

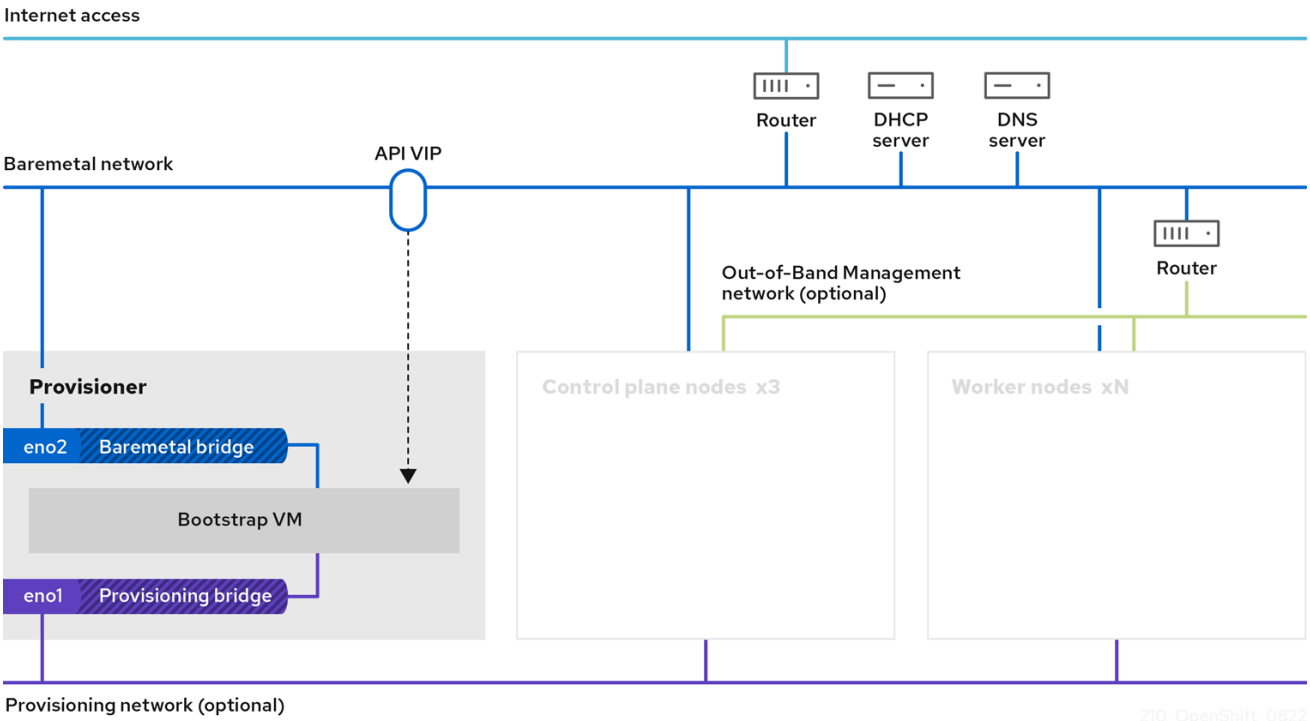
関連情報

- オプション: 非接続クラスターの NTP 設定

- ネットワークタイムプロトコル (NTP)

16.3.4. ネットワークの設定

インストールの前に、プロビジョナーノードでネットワークを設定する必要があります。インストーラーによってプロビジョニングされたクラスターは、ベアメタルブリッジとネットワーク、およびオプションのプロビジョニングブリッジとネットワークを使用してデプロイされます。



注記

Web コンソールからネットワークを設定することもできます。

手順

1. ベアメタルネットワークの NIC 名をエクスポートします。

```
$ export PUB_CONN=<baremetal_nic_name>
```

2. ベアメタルネットワークを設定します。

注記

これらの手順を実行した後、SSH 接続が切断される場合があります。

```
$ sudo nohup bash -c "
    nmcli con down \"$PUB_CONN\"
    nmcli con delete \"$PUB_CONN\"
    # RHEL 8.1 appends the word \"System\" in front of the connection, delete in case it exists
    nmcli con down \"System $PUB_CONN\"
    nmcli con delete \"System $PUB_CONN\"
    nmcli connection add ifname baremetal type bridge con-name baremetal bridge.stp no
```

```
    nmcli con add type bridge-slave ifname \"$PUB_CONN\" master baremetal
    pkill dhclient;dhclient baremetal
"
```

3. オプション: プロビジョニングネットワークを使用してデプロイする場合は、プロビジョニングネットワークの NIC 名をエクスポートします。

```
$ export PROV_CONN=<prov_nic_name>
```

4. オプション: プロビジョニングネットワークを使用してデプロイする場合は、プロビジョニングネットワークを設定します。

```
$ sudo nohup bash -c "
    nmcli con down \"$PROV_CONN\"
    nmcli con delete \"$PROV_CONN\"
    nmcli connection add ifname provisioning type bridge con-name provisioning
    nmcli con add type bridge-slave ifname \"$PROV_CONN\" master provisioning
    nmcli connection modify provisioning ipv6.addresses fd00:1101::1/64 ipv6.method manual
    nmcli con down provisioning
    nmcli con up provisioning
"
```

注記

これらのステップの実行後に ssh 接続が切断される可能性があります。

IPv6 アドレスには、ベアメタルネットワーク経由でルーティング可能でない限り、任意のアドレスを使用できます。

IPv6 アドレスを使用する場合に UEFI PXE 設定が有効にされており、UEFI PXE 設定が IPv6 プロトコルに設定されていることを確認します。

5. オプション: プロビジョニングネットワークを使用してデプロイする場合は、プロビジョニングネットワーク接続で IPv4 アドレスを設定します。

```
$ nmcli connection modify provisioning ipv4.addresses 172.22.0.254/24 ipv4.method manual
```

6. **provisioner** ノードに対して再度 **ssh** を実行します (必要な場合)。

```
# ssh kni@provisioner.<cluster-name>.<domain>
```

7. 接続ブリッジが適切に作成されていることを確認します。

```
$ sudo nmcli con show
```

```
NAME               UUID                                  TYPE      DEVICE
baremetal          4d5133a5-8351-4bb9-bfd4-3af264801530  bridge    baremetal
provisioning       43942805-017f-4d7d-a2c2-7cb3324482ed  bridge    provisioning
virbr0             d9bca40f-eee1-410b-8879-a2d4bb0465e7  bridge    virbr0
bridge-slave-eno1  76a8ed50-c7e5-4999-b4f6-6d9014dd0812  ethernet  eno1
bridge-slave-eno2  f31c3353-54b7-48de-893a-02d2b34c4736  ethernet  eno2
```

16.3.5. サブネット間の通信を確立する

一般的な OpenShift Container Platform クラスターのセットアップでは、コントロールプレーンとコンピュートノードを含むすべてのノードが同じネットワーク内に存在します。ただし、エッジコンピューティングのシナリオでは、コンピュートノードをエッジの近くに配置することが有益な場合があります。その場合、コントロールプレーンやローカルコンピュートノードが使用するサブネットとは異なるネットワークセグメントまたはサブネットをリモートノードに使用することもよくあります。このようなセットアップにより、エッジのレイテンシーが減少し、拡張性が向上します。

OpenShift Container Platform をインストールする前に、リモートノードを含むエッジサブネットがコントロールプレーンノードを含むサブネットに到達し、コントロールプレーンからのトラフィックも受信できるように、ネットワークを適切に設定する必要があります。

デフォルトのロードバランサーの代わりにユーザー管理のロードバランサーを設定することで、同じサブネットまたは複数のサブネットでコントロールプレーンノードを実行できます。複数のサブネット環境を使用すると、ハードウェア障害やネットワーク停止によって OpenShift Container Platform クラスターが失敗するリスクを軽減できます。詳細は、「ユーザー管理ロードバランサーのサービス」および「ユーザー管理ロードバランサーの設定」を参照してください。

複数のサブネット環境でコントロールプレーンノードを実行するには、次の主要なタスクを完了する必要があります。

- **install-config.yaml** ファイルの **loadBalancer.type** パラメーターで **UserManaged** を指定して、デフォルトのロードバランサーの代わりにユーザー管理のロードバランサーを設定します。
- **install-config.yaml** ファイルの **ingressVIPs** および **apiVIPs** パラメーターでユーザー管理のロードバランサーのアドレスを設定します。
- 複数のサブネットの Classless Inter-Domain Routing (CIDR) とユーザー管理のロードバランサーの IP アドレスを、**install-config.yaml** ファイルの **networking.machineNetworks** パラメーターに追加します。

注記

複数のサブネットを持つクラスターをデプロイするには、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディアを使用する必要があります。

この手順では、2 番目のサブネットにあるリモートコンピュートノードが 1 番目のサブネットにあるコントロールプレーンノードと効果的に通信できるようにするために必要なネットワーク設定と、1 番目のサブネットにあるコントロールプレーンノードが 2 番目のサブネットにあるリモートコンピュートノードと効果的に通信できるようにするために必要なネットワーク設定について詳しく説明します。

この手順では、クラスターは 2 つのサブネットにまたがります。

- 1 番目のサブネット (**10.0.0.0**) には、コントロールプレーンとローカルコンピュートノードが含まれています。
- 2 番目のサブネット (**192.168.0.0**) には、エッジコンピュートノードが含まれています。

手順

1. 1 番目のサブネットが 2 番目のサブネットと通信するように設定します。
   a. 次のコマンドを実行して、コントロールプレーンノードに **root** としてログインします。

```
$ sudo su -
```

b. 次のコマンドを実行して、ネットワークインターフェイスの名前を取得します。

```
# nmcli dev status
```

c. 次のコマンドを実行して、ゲートウェイを経由する 2 番目のサブネット (**192.168.0.0**) へのルートを追加します。

```
# nmcli connection modify <interface_name> +ipv4.routes "192.168.0.0/24 via <gateway>"
```

<interface_name> をインターフェイス名に置き換えます。**<gateway>** を実際のゲートウェイの IP アドレスに置き換えます。

例

```
# nmcli connection modify eth0 +ipv4.routes "192.168.0.0/24 via 192.168.0.1"
```

d. 次のコマンドを実行して変更を適用します。

```
# nmcli connection up <interface_name>
```

<interface_name> をインターフェイス名に置き換えます。

e. ルーティングテーブルを検証して、ルートが正常に追加されたことを確認します。

```
# ip route
```

f. 1 番目のサブネットの各コントロールプレーンノードに対して前の手順を繰り返します。

注記

コマンドは、実際のインターフェイス名とゲートウェイに合わせて調整してください。

2. 1 番目のサブネットと通信するように 2 番目のサブネットを設定します。

a. 次のコマンドを実行して、リモートコンピュートノードに **root** としてログインします。

```
$ sudo su -
```

b. 次のコマンドを実行して、ネットワークインターフェイスの名前を取得します。

```
# nmcli dev status
```

c. 次のコマンドを実行して、ゲートウェイを経由する最初のサブネット (**10.0.0.0**) へのルートを追加します。

```
# nmcli connection modify <interface_name> +ipv4.routes "10.0.0.0/24 via <gateway>"
```

<interface_name> をインターフェイス名に置き換えます。**<gateway>** を実際のゲートウェイの IP アドレスに置き換えます。

例

```
# nmcli connection modify eth0 +ipv4.routes "10.0.0.0/24 via 10.0.0.1"
```

d. 次のコマンドを実行して変更を適用します。

```
# nmcli connection up <interface_name>
```

<interface_name> をインターフェイス名に置き換えます。

e. 次のコマンドを実行して、ルーティングテーブルを検証し、ルートが正常に追加されたことを確認します。

```
# ip route
```

f. 2 番目のサブネット内の各コンピュートノードに対して、上記の手順を繰り返します。

注記

コマンドは、実際のインターフェイス名とゲートウェイに合わせて調整してください。

3. ネットワークを設定したら、接続をテストして、リモートノードがコントロールプレーンノードに到達できること、およびコントロールプレーンノードがリモートノードに到達できることを確認します。

a. 1 番目のサブネットのコントロールプレーンノードから、次のコマンドを実行して、2 番目のサブネットのリモートノードに ping を実行します。

```
$ ping <remote_node_ip_address>
```

ping が成功した場合は、1 番目のサブネットのコントロールプレーンノードが 2 番目のサブネットのリモートノードに到達できています。応答がない場合は、ネットワーク設定を確認し、ノードに対して手順を繰り返します。

b. 2 番目のサブネットのリモートノードから、次のコマンドを実行して、1 番目のサブネットのコントロールプレーンノードに ping を実行します。

```
$ ping <control_plane_node_ip_address>
```

ping が成功した場合は、2 番目のサブネットのリモートコンピュートノードが 1 番目のサブネットのコントロールプレーンに到達できています。応答がない場合は、ネットワーク設定を確認し、ノードに対して手順を繰り返します。

16.3.6. OpenShift Container Platform インストーラーの取得

インストールプログラムの **stable-4.x** バージョンと選択したアーキテクチャーを使用して、OpenShift Container Platform の一般公開の安定バージョンをデプロイします。

```
$ export VERSION=stable-4.16
```

```
$ export RELEASE_ARCH=<architecture>
```

```
$ export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-v4/$RELEASE_ARCH/clients/ocp/$VERSION/release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
```

16.3.7. OpenShift Container Platform インストールのデプロイメント

インストーラーを取得したら、インストーラーを展開します。

手順

1. 環境変数を設定します。

```
$ export cmd=openshift-baremetal-install
```

```
$ export pullsecret_file=~/pull-secret.txt
```

```
$ export extract_dir=$(pwd)
```

2. **oc** バイナリーを取得します。

```
$ curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/openshift-client-linux.tar.gz | tar zxvf - oc
```

3. インストーラーを実行します。

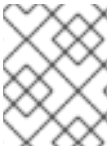
```
$ sudo cp oc /usr/local/bin
```

```
$ oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to "${extract_dir}" ${RELEASE_IMAGE}
```

```
$ sudo cp openshift-baremetal-install /usr/local/bin
```

16.3.8. オプション: RHCOS イメージキャッシュの作成

イメージキャッシングを使用するには、ブートストラップ VM がクラスターノードをプロビジョニングするために使用する Red Hat Enterprise Linux CoreOS(RHCOS) イメージをダウンロードする必要があります。イメージのキャッシュはオプションですが、帯域幅が制限されているネットワークでインストールプログラムを実行する場合に特に便利です。



注記

正しいイメージがリリースペイロードにあるため、インストールプログラムは、**clusterOSImage** RHCOS イメージを必要としなくなりました。

帯域幅が制限されたネットワークでインストールプログラムを実行していて、RHCOS イメージのダウンロードに 15～20 分以上かかる場合、インストールプログラムはタイムアウトになります。このような場合、Web サーバーでイメージをキャッシュすることができます。

警告

HTTPD サーバーに対して TLS を有効にする場合、ルート証明書がクライアントによって信頼された機関によって署名されていることを確認し、OpenShift Container Platform ハブおよびスポーククラスターと HTTPD サーバー間の信頼された証明書チェーンを検証する必要があります。信頼されていない証明書で設定されたサーバーを使用すると、イメージがイメージ作成サービスにダウンロードされなくなります。信頼されていない HTTPS サーバーの使用はサポートされていません。

イメージを含むコンテナーをインストールします。

手順

1. **podman** をインストールします。

```
$ sudo dnf install -y podman
```

2. RHCOS イメージのキャッシュに使用されるファイアウォールのポート **8080** を開きます。

```
$ sudo firewall-cmd --add-port=8080/tcp --zone=public --permanent
```

```
$ sudo firewall-cmd --reload
```

3. **bootstraposimage** を保存するディレクトリーを作成します。

```
$ mkdir /home/kni/rhcos_image_cache
```

4. 新規に作成されたディレクトリーに適切な SELinux コンテキストを設定します。

```
$ sudo semanage fcontext -a -t httpd_sys_content_t "/home/kni/rhcos_image_cache(/.*)?"
```

```
$ sudo restorecon -Rv /home/kni/rhcos_image_cache/
```

5. インストールプログラムがブートストラップ VM にデプロイする RHCOS イメージの URI を取得します。

```
$ export RHCOS_QEMU_URI=$(/usr/local/bin/openshift-baremetal-install coreos print-stream-json | jq -r --arg ARCH "$(arch)" '.architectures[$ARCH].artifacts.qemu.formats["qcow2.gz"].disk.location')
```

6. インストールプログラムがブートストラップ VM にデプロイするイメージの名前を取得します。

```
$ export RHCOS_QEMU_NAME=${RHCOS_QEMU_URI##*/}
```

7. ブートストラップ仮想マシンにデプロイされる RHCOS イメージの SHA ハッシュを取得します。

```
$ export RHCOS_QEMU_UNCOMPRESSED_SHA256=$(/usr/local/bin/openshift-baremetal-
install coreos print-stream-json | jq -r --arg ARCH "$(arch)"
'.architectures[$ARCH].artifacts.qemu.formats["qcow2.gz"].disk["uncompressed-sha256"]')
```

8. イメージをダウンロードして、**/home/kni/rhcos_image_cache** ディレクトリーに配置します。

```
$ curl -L ${RHCOS_QEMU_URI} -o
/home/kni/rhcos_image_cache/${RHCOS_QEMU_NAME}
```

9. 新しいファイルの SELinux タイプが **httpd_sys_content_t** であることを確認します。

```
$ ls -Z /home/kni/rhcos_image_cache
```

10. Pod を作成します。

```
$ podman run -d --name rhcos_image_cache \ 1
-v /home/kni/rhcos_image_cache:/var/www/html \
-p 8080:8080/tcp \
registry.access.redhat.com/ubi9/httpd-24
```

1 **rhcos_image_cache** という名前のキャッシング Web サーバーを作成します。この Pod は、デプロイメント用に **install-config.yaml** ファイルの **bootstrapOSImage** イメージを提供します。

11. **bootstrapOSImage** 設定を生成します。

```
$ export BAREMETAL_IP=$(ip addr show dev baremetal | awk '/inet /{print $2}' | cut -d"/" -f1)
```

```
$ export
BOOTSTRAP_OS_IMAGE="http://${BAREMETAL_IP}:8080/${RHCOS_QEMU_NAME}?
sha256=${RHCOS_QEMU_UNCOMPRESSED_SHA256}"
```

```
$ echo "    bootstrapOSImage=${BOOTSTRAP_OS_IMAGE}"
```

12. **platform.baremetal** 下の **install-config.yaml** ファイルに必要な設定を追加します。

```
platform:
  baremetal:
    bootstrapOSImage: <bootstrap_os_image> 1
```

1 **<bootstrap_os_image>** を **$BOOTSTRAP_OS_IMAGE** の値に置き換えます。

詳細は、Configuring the install-config.yaml file セクションを参照してください。

16.3.9. ユーザー管理ロードバランサーのサービス

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。

重要

ユーザー管理ロードバランサーの設定は、ベンダーのロードバランサーによって異なります。

このセクションの情報と例は、ガイドラインのみを目的としています。ベンダーのロードバランサーに関する詳細は、ベンダーのドキュメントを参照してください。

Red Hat は、ユーザー管理ロードバランサーに対して次のサービスをサポートしています。

- Ingress Controller
- OpenShift API
- OpenShift MachineConfig API

ユーザー管理ロードバランサーに対して、これらのサービスの 1 つを設定するか、すべてを設定するかを選択できます。一般的な設定オプションは、Ingress Controller サービスのみを設定することです。次の図は、各サービスの詳細を示しています。

図16.1 OpenShift Container Platform 環境で動作する Ingress Controller を示すネットワークワークフローの例

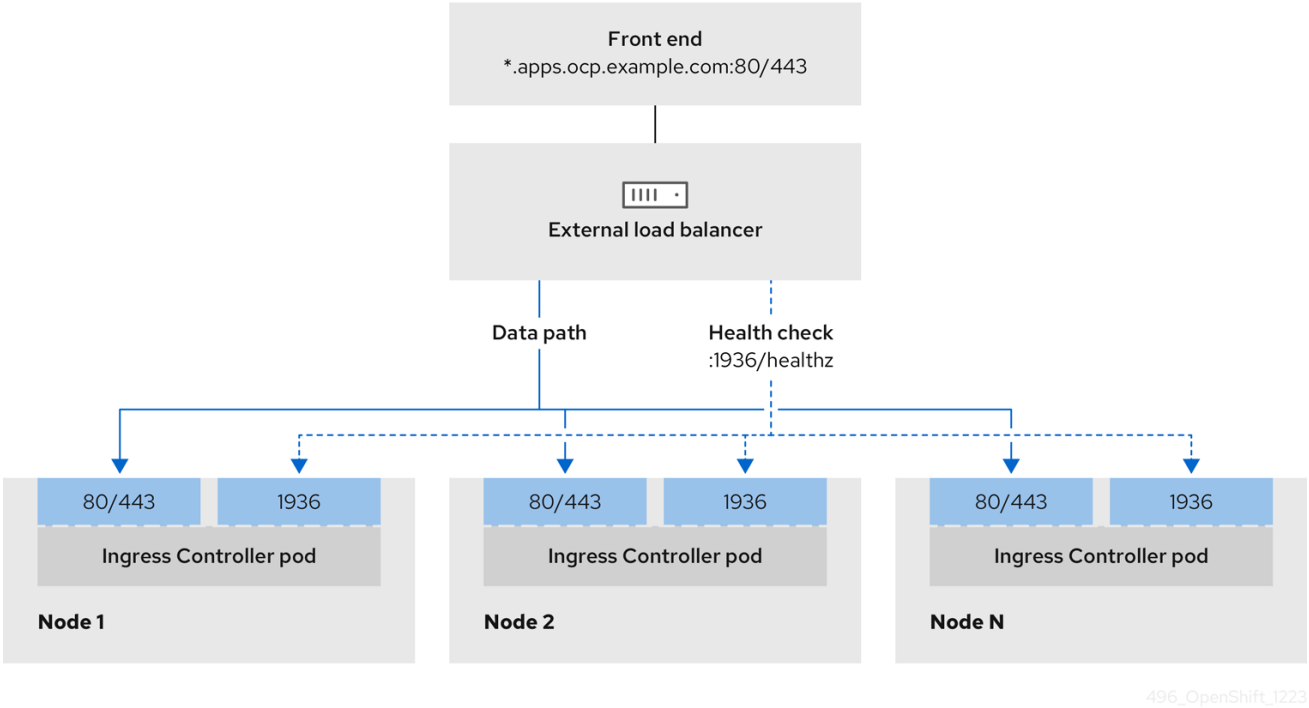


図16.2 OpenShift Container Platform 環境で動作する OpenShift API を示すネットワークワークフローの例

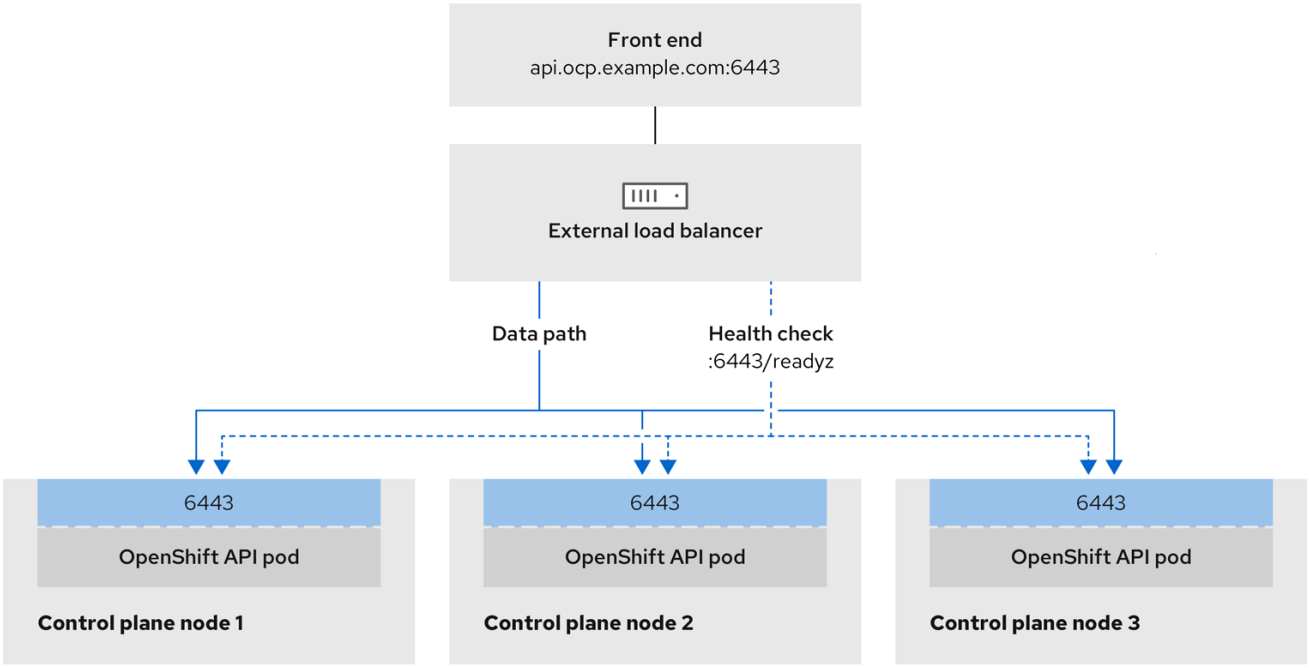
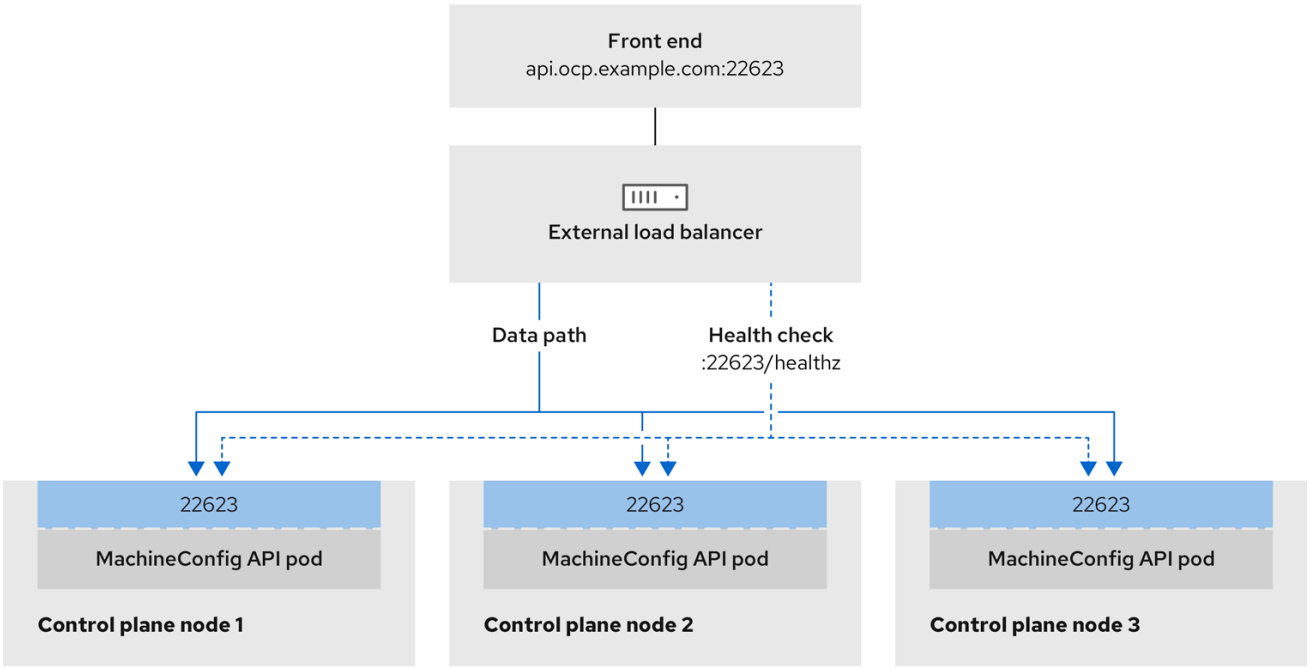


図16.3 OpenShift Container Platform 環境で動作する OpenShift MachineConfig API を示すネットワークワークフローの例



ユーザー管理ロードバランサーでは、次の設定オプションがサポートされています。

- ノードセレクターを使用して、Ingress Controller を特定のノードのセットにマッピングします。このセットの各ノードに静的 IP アドレスを割り当てるか、Dynamic Host Configuration Protocol (DHCP) から同じ IP アドレスを受け取るように各ノードを設定する必要があります。インフラストラクチャーノードは通常、このタイプの設定を受け取ります。

- サブネット上のすべての IP アドレスをターゲットにします。この設定では、ロードバランサーターゲットを再設定せずにネットワーク内でノードを作成および破棄できるため、メンテナンスオーバーヘッドを削減できます。**/27** や **/28** などの小規模なネットワーク上に設定されたマシンを使用して Ingress Pod をデプロイする場合、ロードバランサーのターゲットを簡素化できます。

  ヒント

  マシン config プールのリソースを確認することで、ネットワーク内に存在するすべての IP アドレスをリスト表示できます。

OpenShift Container Platform クラスターのユーザー管理ロードバランサーを設定する前に、以下の情報を考慮してください。

- フロントエンド IP アドレスの場合、フロントエンド IP アドレス、Ingress Controller のロードバランサー、および API ロードバランサーに同じ IP アドレスを使用できます。この機能については、ベンダーのドキュメントを確認してください。

- バックエンド IP アドレスの場合、ユーザー管理ロードバランサーの有効期間中に OpenShift Container Platform コントロールプレーンノードの IP アドレスが変更されないことを確認します。次のいずれかのアクションを実行すると、これを実現できます。

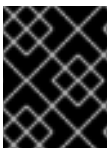
  - 各コントロールプレーンノードに静的 IP アドレスを割り当てます。

  - ノードが DHCP リースを要求するたびに、DHCP から同じ IP アドレスを受信するように各ノードを設定します。ベンダーによっては、DHCP リースは IP 予約または静的 DHCP 割り当ての形式になる場合があります。

- Ingress Controller バックエンドサービスのユーザー管理ロードバランサーで Ingress Controller を実行する各ノードを手動で定義します。たとえば、Ingress Controller が未定義のノードに移動すると、接続が停止する可能性があります。

16.3.9.1. ユーザー管理ロードバランサーの設定

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。



重要

ユーザー管理ロードバランサーを設定する前に、ユーザー管理ロードバランサーのサービス" セクションを必ずお読みください。

ユーザー管理ロードバランサー用に設定するサービスに適用される次の前提条件をお読みください。

注記

クラスター上で実行される MetalLB は、ユーザー管理ロードバランサーとして機能します。

OpenShift API の前提条件

- フロントエンド IP アドレスを定義している。

- TCP ポート 6443 および 22623 は、ロードバランサーのフロントエンド IP アドレスで公開されている。以下の項目を確認します。
  - ポート 6443 が OpenShift API サービスにアクセスできる。
  - ポート 22623 が Ignition 起動設定をノードに提供できる。
- フロントエンド IP アドレスとポート 6443 へは、OpenShift Container Platform クラスターの外部の場所にいるシステムのすべてのユーザーがアクセスできる。
- フロントエンド IP アドレスとポート 22623 は、OpenShift Container Platform ノードからのみ到達できる。
- ロードバランサーバックエンドは、ポート 6443 および 22623 の OpenShift Container Platform コントロールプレーンノードと通信できる。

Ingress Controller の前提条件

- フロントエンド IP アドレスを定義している。
- TCP ポート 443 および 80 はロードバランサーのフロントエンド IP アドレスで公開されている。
- フロントエンドの IP アドレス、ポート 80、ポート 443 へは、OpenShift Container Platform クラスターの外部の場所にあるシステムの全ユーザーがアクセスできる。
- フロントエンドの IP アドレス、ポート 80、ポート 443 は、OpenShift Container Platform クラスターで動作するすべてのノードから到達できる。
- ロードバランサーバックエンドは、ポート 80、443、および 1936 で Ingress Controller を実行する OpenShift Container Platform ノードと通信できる。

ヘルスチェック URL 仕様の前提条件

ほとんどのロードバランサーは、サービスが使用可能か使用不可かを判断するヘルスチェック URL を指定して設定できまうｓ．OpenShift Container Platform は、OpenShift API、Machine Configuration API、および Ingress Controller バックエンドサービスのこれらのヘルスチェックを提供します。

次の例は、前にリスト表示したバックエンドサービスのヘルスチェック仕様を示しています。

Kubernetes API ヘルスチェック仕様の例

```
Path: HTTPS:6443/readyz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Machine Config API ヘルスチェック仕様の例

```
Path: HTTPS:22623/healthz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Ingress Controller のヘルスチェック仕様の例

```
Path: HTTP:1936/healthz/ready
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 5
Interval: 10
```

手順

1. HAProxy Ingress Controller を設定して、ポート 6443、22623、443、および 80 でロードバランサーからクラスターへのアクセスを有効化できるようにします。必要に応じて、HAProxy 設定で単一のサブネットの IP アドレスまたは複数のサブネットの IP アドレスを指定できます。

   #### 1つのサブネットをリストした HAProxy 設定の例

   ```
   # ...
   listen my-cluster-api-6443
       bind 192.168.1.100:6443
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /readyz
     http-check expect status 200
       server my-cluster-master-2 192.168.1.101:6443 check inter 10s rise 2 fall 2
       server my-cluster-master-0 192.168.1.102:6443 check inter 10s rise 2 fall 2
       server my-cluster-master-1 192.168.1.103:6443 check inter 10s rise 2 fall 2

   listen my-cluster-machine-config-api-22623
       bind 192.168.1.100:22623
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /healthz
     http-check expect status 200
       server my-cluster-master-2 192.168.1.101:22623 check inter 10s rise 2 fall 2
       server my-cluster-master-0 192.168.1.102:22623 check inter 10s rise 2 fall 2
       server my-cluster-master-1 192.168.1.103:22623 check inter 10s rise 2 fall 2

   listen my-cluster-apps-443
       bind 192.168.1.100:443
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /healthz/ready
     http-check expect status 200
       server my-cluster-worker-0 192.168.1.111:443 check port 1936 inter 10s rise 2 fall 2
       server my-cluster-worker-1 192.168.1.112:443 check port 1936 inter 10s rise 2 fall 2
       server my-cluster-worker-2 192.168.1.113:443 check port 1936 inter 10s rise 2 fall 2

   listen my-cluster-apps-80
     bind 192.168.1.100:80
   ```

```
  mode tcp
  balance roundrobin
 option httpchk
 http-check connect
 http-check send meth GET uri /healthz/ready
 http-check expect status 200
   server my-cluster-worker-0 192.168.1.111:80 check port 1936 inter 10s rise 2 fall 2
   server my-cluster-worker-1 192.168.1.112:80 check port 1936 inter 10s rise 2 fall 2
   server my-cluster-worker-2 192.168.1.113:80 check port 1936 inter 10s rise 2 fall 2
# ...
```

複数のサブネットをリストした HAProxy 設定の例

```
# ...
listen api-server-6443
   bind *:6443
   mode tcp
     server master-00 192.168.83.89:6443 check inter 1s
     server master-01 192.168.84.90:6443 check inter 1s
     server master-02 192.168.85.99:6443 check inter 1s
     server bootstrap 192.168.80.89:6443 check inter 1s

listen machine-config-server-22623
   bind *:22623
   mode tcp
     server master-00 192.168.83.89:22623 check inter 1s
     server master-01 192.168.84.90:22623 check inter 1s
     server master-02 192.168.85.99:22623 check inter 1s
     server bootstrap 192.168.80.89:22623 check inter 1s

listen ingress-router-80
   bind *:80
   mode tcp
   balance source
     server worker-00 192.168.83.100:80 check inter 1s
     server worker-01 192.168.83.101:80 check inter 1s

listen ingress-router-443
   bind *:443
   mode tcp
   balance source
     server worker-00 192.168.83.100:443 check inter 1s
     server worker-01 192.168.83.101:443 check inter 1s

listen ironic-api-6385
   bind *:6385
   mode tcp
   balance source
     server master-00 192.168.83.89:6385 check inter 1s
     server master-01 192.168.84.90:6385 check inter 1s
     server master-02 192.168.85.99:6385 check inter 1s
     server bootstrap 192.168.80.89:6385 check inter 1s

listen inspector-api-5050
   bind *:5050
   mode tcp
```

```
  balance source
    server master-00 192.168.83.89:5050 check inter 1s
    server master-01 192.168.84.90:5050 check inter 1s
    server master-02 192.168.85.99:5050 check inter 1s
    server bootstrap 192.168.80.89:5050 check inter 1s
# ...
```

2. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーとそのリソースが動作していることを確認します。

   a. 次のコマンドを実行して応答を観察し、クラスターマシン設定 API が Kubernetes API サーバーリソースにアクセスできることを確認します。

      ```
      $ curl https://<loadbalancer_ip_address>:6443/version --insecure
      ```

      設定が正しい場合は、応答として JSON オブジェクトを受信します。

      ```
      {
        "major": "1",
        "minor": "11+",
        "gitVersion": "v1.11.0+ad103ed",
        "gitCommit": "ad103ed",
        "gitTreeState": "clean",
        "buildDate": "2019-01-09T06:44:10Z",
        "goVersion": "go1.10.3",
        "compiler": "gc",
        "platform": "linux/amd64"
      }
      ```

   b. 次のコマンドを実行して出力を確認し、クラスターマシン設定 API がマシン設定サーバーリソースからアクセスできることを確認します。

      ```
      $ curl -v https://<loadbalancer_ip_address>:22623/healthz --insecure
      ```

      設定が正しい場合、コマンドの出力には次の応答が表示されます。

      ```
      HTTP/1.1 200 OK
      Content-Length: 0
      ```

   c. 次のコマンドを実行して出力を確認し、コントローラーがポート 80 の Ingress Controller リソースにアクセスできることを確認します。

      ```
      $ curl -I -L -H "Host: console-openshift-console.apps.<cluster_name>.<base_domain>"
      http://<load_balancer_front_end_IP_address>
      ```

      設定が正しい場合、コマンドの出力には次の応答が表示されます。

      ```
      HTTP/1.1 302 Found
      content-length: 0
      location: https://console-openshift-console.apps.ocp4.private.opequon.net/
      cache-control: no-cache
      ```

   d. 次のコマンドを実行して出力を確認し、コントローラーがポート 443 の Ingress Controller リソースにアクセスできることを確認します。

```
$ curl -I -L --insecure --resolve console-openshift-console.apps.<cluster_name>.
<base_domain>:443:<Load Balancer Front End IP Address> https://console-openshift-
console.apps.<cluster_name>.<base_domain>
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

3. ユーザー管理ロードバランサーのフロントエンド IP アドレスをターゲットにするようにクラスターの DNS レコードを設定します。ロードバランサー経由で、クラスター API およびアプリケーションの DNS サーバーのレコードを更新する必要があります。

   変更された DNS レコードの例

   ```
   <load_balancer_ip_address>  A  api.<cluster_name>.<base_domain>
   A record pointing to Load Balancer Front End
   ```

   ```
   <load_balancer_ip_address>   A apps.<cluster_name>.<base_domain>
   A record pointing to Load Balancer Front End
   ```

   重要

   DNS の伝播では、各 DNS レコードが使用可能になるまでに時間がかかる場合があります。各レコードを検証する前に、各 DNS レコードが伝播されることを確認してください。

4. OpenShift Container Platform クラスターでユーザー管理ロードバランサーを使用するには、クラスターの **install-config.yaml** ファイルで次の設定を指定する必要があります。

   ```
   # ...
   platform:
     baremetal:
       loadBalancer:
         type: UserManaged 1
         apiVIPs:
         - <api_ip> 2
   ```

```
    ingressVIPs:
    - <ingress_ip> 3
# ...
```

1 クラスターのユーザー管理ロードバランサーを指定するには、**type** パラメーターに **UserManaged** を設定します。パラメーターのデフォルトは **OpenShiftManagedDefault** で、これはデフォルトの内部ロードバランサーを示します。**openshift-kni-infra** namespace で定義されたサービスの場合、ユーザー管理ロードバランサーは **coredns** サービスをクラスター内の Pod にデプロイできますが、**keepalived** および **haproxy** サービスは無視します。

2 ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。Kubernetes API がユーザー管理ロードバランサーと通信できるように、ユーザー管理ロードバランサーのパブリック IP アドレスを指定します。

3 ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。ユーザー管理ロードバランサーのパブリック IP アドレスを指定して、ユーザー管理ロードバランサーがクラスターの Ingress トラフィックを管理できるようにします。

検証

1. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーと DNS レコード設定が動作していることを確認します。

   a. 次のコマンドを実行して出力を確認し、クラスター API にアクセスできることを確認します。

   ```
   $ curl https://api.<cluster_name>.<base_domain>:6443/version --insecure
   ```

   設定が正しい場合は、応答として JSON オブジェクトを受信します。

   ```
   {
     "major": "1",
     "minor": "11+",
     "gitVersion": "v1.11.0+ad103ed",
     "gitCommit": "ad103ed",
     "gitTreeState": "clean",
     "buildDate": "2019-01-09T06:44:10Z",
     "goVersion": "go1.10.3",
     "compiler": "gc",
     "platform": "linux/amd64"
     }
   ```

   b. 次のコマンドを実行して出力を確認し、クラスターマシン設定にアクセスできることを確認します。

   ```
   $ curl -v https://api.<cluster_name>.<base_domain>:22623/healthz --insecure
   ```

   設定が正しい場合、コマンドの出力には次の応答が表示されます。

   ```
   HTTP/1.1 200 OK
   Content-Length: 0
   ```

c. 以下のコマンドを実行して出力を確認し、ポートで各クラスターアプリケーションにアクセスできることを確認します。

```
$ curl http://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.<cluster-name>.<base domain>/
cache-control: no-cacheHTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=39HoZgztDnzjJkq/JuLJMeoKNXlfiVv2YgZc09c3TBOBU4NI6kDXaJH1LdicNhN1UsQ
Wzon4Dor9GWGfopaTEQ==; Path=/; Secure
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Tue, 17 Nov 2020 08:42:10 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=9b714eb87e93cf34853e87a92d6894be; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

d. 次のコマンドを実行して出力を確認し、ポート 443 で各クラスターアプリケーションにアクセスできることを確認します。

```
$ curl https://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

16.3.10. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は **NetworkManager** で設定されま

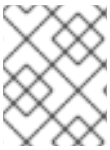
す。デフォルトでは、マシンは DHCP を通じてホスト名を取得します。ホスト名が DHCP によって提供されない場合や、カーネル引数または別の方法により静的に設定される場合、ホスト名は逆引き DNS ルックアップにより取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスがこれより前に起動して、ホスト名を **localhost** や同様の名前として検出する可能性があります。このようなホスト名割り当ての遅延は、DHCP を使用して各クラスターノードにホスト名を提供することで回避できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

16.3.11. install-config.yaml ファイルの設定

16.3.11.1. install-config.yaml ファイルの設定

install-config.yaml ファイルには、追加の詳細情報が必要です。ほとんどの情報は、インストールプログラムと結果として作成されるクラスターに、完全に管理できる使用可能なハードウェアについて十分に説明しています。



注記

正しいイメージがリリースペイロードにあるため、インストールプログラムは、**clusterOSImage** RHCOS イメージを必要としなくなりました。

1. **install-config.yaml** を設定します。**pullSecret**、**sshKey** など、環境に合わせて適切な変数を変更します。

```
apiVersion: v1
baseDomain: <domain>
metadata:
  name: <cluster_name>
networking:
  machineNetwork:
  - cidr: <public_cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2 (1)
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIPs:
      - <api_ip>
    ingressVIPs:
      - <wildcard_ip>
    provisioningNetworkCIDR: <CIDR>
    bootstrapExternalStaticIP: <bootstrap_static_ip_address> (2)
    bootstrapExternalStaticGateway: <bootstrap_static_gateway> (3)
    bootstrapExternalStaticDNS: <bootstrap_static_dns> (4)
    hosts:
      - name: openshift-master-0
```

```
      role: master
      bmc:
        address: ipmi://<out_of_band_ip> 5
        username: <user>
        password: <password>
      bootMACAddress: <NIC1_mac_address>
      rootDeviceHints:
       deviceName: "<installation_disk_drive_path>" 6
    - name: <openshift_master_1>
      role: master
      bmc:
        address: ipmi://<out_of_band_ip>
        username: <user>
        password: <password>
      bootMACAddress: <NIC1_mac_address>
      rootDeviceHints:
       deviceName: "<installation_disk_drive_path>"
    - name: <openshift_master_2>
      role: master
      bmc:
        address: ipmi://<out_of_band_ip>
        username: <user>
        password: <password>
      bootMACAddress: <NIC1_mac_address>
      rootDeviceHints:
       deviceName: "<installation_disk_drive_path>"
    - name: <openshift_worker_0>
      role: worker
      bmc:
        address: ipmi://<out_of_band_ip>
        username: <user>
        password: <password>
      bootMACAddress: <NIC1_mac_address>
    - name: <openshift_worker_1>
      role: worker
      bmc:
        address: ipmi://<out_of_band_ip>
        username: <user>
        password: <password>
      bootMACAddress: <NIC1_mac_address>
      rootDeviceHints:
       deviceName: "<installation_disk_drive_path>"
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

1 OpenShift Container Platform クラスターの一部であるコンピュートノードの数に基づいて、コンピュートマシンをスケーリングします。**replicas** 値の有効なオプションは **0** で、**2** 以上の整数です。3 ノードクラスターのみが含まれる 3 ノードクラスターをデプロイするには、レプリカ数を **0** に設定します。3 ノードクラスターは、テスト、開発、本番に使用できる、より小さく、よりリソース効率の良いクラスターです。コンピュートノードが 1 つだけのクラスターをインストールすることはできません。

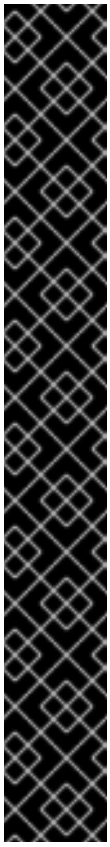
2 静的 IP アドレスを使用してクラスターをデプロイする場合、ベアメタルネットワークに DHCP サーバーがない場合は、**bootstrapExternalStaticIP** 設定を設定して、ブートストラップ VM の静的 IP アドレスを指定する必要があります。

3 静的 IP アドレスを使用してクラスターをデプロイする場合、ベアメタルネットワークに DHCP サーバーがない場合は、**bootstrapExternalStaticGateway** 設定を設定して、ブー

4 静的 IP アドレスを使用してクラスターをデプロイする場合、ベアメタルネットワークに DHCP サーバーがない場合は、**bootstrapExternalStaticDNS** 設定を設定して、ブートストラップ仮想マシンの DNS アドレスを指定する必要があります。

5 その他のオプションについては、BMC アドレス指定のセクションを参照してください。

6 インストールディスクドライブへのパスを設定するには、ディスクのカーネル名を入力します。たとえば、**/dev/sda** です。



重要

ディスクの検出順序は保証されていないため、複数のディスクを備えたマシンの起動オプションによってディスクのカーネル名が変わる可能性があります。たとえば、**/dev/sda** は **/dev/sdb** になり、その逆も同様です。この問題を回避するには、ディスクの World Wide Name (WWN) や **/dev/disk/by-path/** などの永続的なディスク属性を使用する必要があります。ストレージの場所への **/dev/disk/by-path/<device_path>** リンクを使用することを推奨します。ディスク WWN を使用するには、**deviceName** パラメーターを **wwnWithExtension** パラメーターに置き換えます。使用するパラメーターに応じて、次のいずれかの値を入力します。

- ディスク名。たとえば、**/dev/sda**、または **/dev/disk/by-path/** です。
- ディスクの WWN。たとえば、**"0x64cd98f04fde100024684cf3034da5c2"** です。ディスク WWN 値が 16 進数値ではなく文字列値として使用されるように、ディスク WWN 値を引用符で囲んで入力してください。

これらの **rootDeviceHints** パラメーター要件を満たさない場合、次のエラーが発生する可能性があります。

```
ironic-inspector inspection failed: No disks satisfied root device hints
```

注記

OpenShift Container Platform 4.12 より前では、クラスターインストールプログラムが、**apiVIP** および **ingressVIP** 設定の IPv4 アドレスまたは IPv6 アドレスのみを受け入れていました。OpenShift Container Platform 4.12 以降では、これらの設定は非推奨です。代わりに、**apiVIPs** および **ingressVIPs** 設定でリスト形式を使用して、IPv4 アドレス、IPv6 アドレス、または両方の IP アドレス形式を指定してください。

2. クラスター設定を保存するディレクトリーを作成します。

```
$ mkdir ~/clusterconfigs
```

3. **install-config.yaml** ファイルを新しいディレクトリーにコピーします。

```
$ cp install-config.yaml ~/clusterconfigs
```

4. OpenShift Container Platform クラスターをインストールする前に、すべてのベアメタルノードの電源がオフになっていることを確認します。

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

5. 以前に試行したデプロイメントにより古いブートストラップリソースが残っている場合は、これを削除します。

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

16.3.11.2. 追加の install-config パラメーター

install-config.yaml ファイルに必要なパラメーター **hosts** パラメーターおよび **bmc** パラメーターについては、以下の表を参照してください。

表16.5 必須パラメーター

パラメーター	デフォルト	説明
baseDomain		クラスターのドメイン名。例: **example.com**
bootMode	**UEFI**	ノードのブートモード。オプションは、**legacy**、**UEFI**、および **UEFISecureBoot** です。**bootMode** が設定されていない場合は、ノードを検査する間に Ironic がこれを設定します。
bootstrapExternalStaticDNS		ブートストラップノードの静的ネットワーク DNS。ベアメタルネットワークに Dynamic Host Configuration Protocol (DHCP) サーバーがない場合に、静的 IP アドレスを使用してクラスターをデプロイする場合は、この値を設定する必要があります。この値を設定しないと、インストールプログラムが **bootstrapExternalStaticGateway** の値を使用します。そのため、ゲートウェイと DNS の IP アドレス値が異なる場合に問題が発生します。
bootstrapExternalStaticIP		ブートストラップ VM の静的 IP アドレス。ベアメタルネットワークに DHCP サーバーがない場合に、静的 IP アドレスを使用してクラスターをデプロイする場合は、この値を設定する必要があります。
bootstrapExternalStaticGateway		ブートストラップ VM のゲートウェイの静的 IP アドレス。ベアメタルネットワークに DHCP サーバーがない場合に、静的 IP アドレスを使用してクラスターをデプロイする場合は、この値を設定する必要があります。

パラメーター	デフォルト	説明
sshKey		**sshKey** 設定には、コントロールプレーンノードとコンピュートノードにアクセスするために必要な、**~/.ssh/id_rsa.pub** ファイル内のキーを含めます。通常、このキーは **provisioner** ノードのキーです。
pullSecret		**pullSecret** 設定には、プロビジョナーノードの準備の際に Install OpenShift on Bare Metal ページからダウンロードしたプルシークレットのコピーが含まれます。
metadata: name:		OpenShift Container Platform クラスターに指定される名前。例: **openshift**
networking: machineNetwork: - cidr:		外部ネットワークの公開 CIDR (Classless Inter-Domain Routing)。例: **10.0.0.0/24** など。
compute: - name: worker		OpenShift Container Platform クラスターでは、ノードがゼロの場合でも、コンピュートノードに名前を付ける必要があります。
compute: replicas: 2		replicas は、OpenShift Container Platform クラスターのコンピュートノードの数を設定します。
controlPlane: name: master		OpenShift Container Platform クラスターでは、コントロールプレーンノードに名前が必要です。
controlPlane: replicas: 3		replicas は、OpenShift Container Platform クラスターの一部として含まれるコントロールプレーンノードの数を設定します。
provisioningNetworkInterface		ベアメタルネットワークに接続されたノード上のネットワークインターフェイス名。OpenShift Container Platform 4.9 以降のリリースのために、NIC の名前を識別するために **provisioningNetworkInterface** 設定を使用する代わりに、Ironic が NIC の IP アドレスを識別できるように **bootMACAddress** 設定を使用します。
defaultMachinePlatform		プラットフォーム設定なしでマシンプールに使用されるデフォルト設定。

パラメーター	デフォルト	説明
apiVIPs		(オプション) Kubernetes API 通信の仮想 IP アドレス。 この設定は、Machine Network からの予約済み IP として **install-config.yaml** ファイルで提供するか、デフォルト名が正しく解決されるように DNS で事前設定する必要があります。**install-config.yaml** ファイルの **apiVIPs** 設定に値を追加するときは、FQDN ではなく仮想 IP アドレスを使用します。デュアルスタックネットワークを使用する場合には、プライマリー IP アドレスは IPv4 ネットワークからのものである必要があります。設定されていない場合、インストールプログラムは **api.<cluster_name>.<base_domain>** を使用して DNS から IP アドレスを取得します。  **注記** OpenShift Container Platform 4.12 より前では、クラスターのインストールプログラムは **apiVIP** 設定の IPv4 アドレスまたは IPv6 アドレスのみを受け入れていました。OpenShift Container Platform 4.12 以降から、**apiVIP** 設定は非推奨になりました。代わりに、**apiVIPs** 設定のリスト形式を使用して、IPv4 アドレス、IPv6 アドレス、または両方の IP アドレス形式を指定します。
disableCertificateVerification	**False**	**redfish** および **redfish-virtualmedia** では、このパラメーターで BMC アドレスを管理する必要があります。BMC アドレスに自己署名証明書を使用する場合は、値が **True** である必要があります。

パラメーター	デフォルト	説明
ingressVIPs		(オプション) Ingress トラフィックの仮想 IP アドレス。 この設定は、Machine Network からの予約済み IP として **install-config.yaml** ファイルで提供するか、デフォルト名が正しく解決されるように DNS で事前設定する必要があります。**install-config.yaml** ファイルの **ingressVIPs** 構成設定に値を追加するときは、FQDN ではなく仮想 IP アドレスを使用してください。デュアルスタックネットワークを使用する場合には、プライマリー IP アドレスは IPv4 ネットワークからのものである必要があります。設定されていない場合、インストールプログラムは **test.apps.<cluster_name>.<base_domain>** を使用して DNS から IP アドレスを取得します。 **注記** OpenShift Container Platform 4.12 より前では、クラスターのインストールプログラムは **ingressVIP** 設定の IPv4 アドレスまたは IPv6 アドレスのみを受け入れていました。OpenShift Container Platform 4.12 以降では、**ingressVIP** 設定は非推奨です。代わりに、**ingressVIPs** 設定のリスト形式を使用して、IPv4 アドレス、IPv6 アドレス、または両方の IP アドレス形式を指定します。

表16.6 オプションのパラメーター

パラメーター	デフォルト	説明
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	プロビジョニングネットワークでノードの IP 範囲を定義します。
provisioningNetworkCIDR	**172.22.0.0/24**	プロビジョニングに使用するネットワークの CIDR。このオプションは、プロビジョニングネットワークでデフォルトのアドレス範囲を使用しない場合に必要です。
clusterProvisioningIP	**provisioningNetworkCIDR** の 3 番目の IP アドレス。	プロビジョニングサービスが実行されるクラスター内の IP アドレス。デフォルトは、プロビジョニングサブネットの 3 番目の IP アドレスに設定されます。例: **172.22.0.3**.
bootstrapProvisioningIP	**provisioningNetworkCIDR** の 2 番目の IP アドレス	インストーラーがコントロールプレーン (マスター) ノードをデプロイしている間にプロビジョニングサービスが実行されるブートストラップ仮想マシンの IP アドレス。デフォルトは、プロビジョニングサブネットの 2 番目の IP アドレスに設定されます。例: **172.22.0.2** または **2620:52:0:1307::2**。
externalBridge	**baremetal**	ベアメタルネットワークに接続されたハイパーバイザーのベアメタルブリッジの名前。

パラメーター	デフォルト	説明
provisioningBridge	**provisioning**	プロビジョニングネットワークに接続されている **provisioner** ホストのプロビジョニングブリッジの名前。
architecture		クラスターのホストアーキテクチャーを定義します。有効な値は **amd64** または **arm64** です。
defaultMachinePlatform		プラットフォーム設定なしでマシンプールに使用されるデフォルト設定。
bootstrapOSImage		ブートストラップノードのデフォルトのオペレーティングシステムイメージを上書きするための URL。URL にはイメージの SHA-256 ハッシュが含まれている必要があります。例: **https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>**。
provisioningNetwork		**provisioningNetwork** 設定は、クラスターがプロビジョニングネットワークを使用するかどうかを決定します。存在する場合、設定はクラスターがネットワークを管理するかどうかも決定します。 **Disabled**: プロビジョニングネットワークの要件を無効にするには、このパラメーターを **Disabled** に設定します。**Disabled** に設定すると、仮想メディアベースのプロビジョニングのみを使用するか、アシステッドインストーラーを使用してクラスターを起動する必要があります。**Disabled**にして、電源管理を使用する場合、BMC はベアメタルネットワークからアクセスできる必要があります。**Disabled** の場合は、プロビジョニングサービスに使用されるベアメタルネットワークで 2 つの IP アドレスを指定する必要があります。 **Managed**: DHCP、TFTP などを含むプロビジョニングネットワークを完全に管理するには、このパラメーターを **Managed**(デフォルト) に設定します。 **Unmanaged**: このパラメーターを **Unmanaged** に設定してプロビジョニングネットワークを引き続き有効にしますが、DHCP の手動設定を行います。仮想メディアのプロビジョニングが推奨されますが、必要に応じて PXE を引き続き利用できます。
httpProxy		このパラメーターを、環境内で使用する適切な HTTP プロキシーに設定します。
httpsProxy		このパラメーターを、環境内で使用する適切な HTTPS プロキシーに設定します。
noProxy		このパラメーターを、環境内のプロキシーの使用に対する例外のリストに設定します。

ホスト
hosts パラメーターは、クラスターのビルドに使用される個別のベアメタルアセットのリストです。

表16.7 ホスト

名前	デフォルト	説明
name		詳細情報に関連付ける **BareMetalHost** リソースの名前。例: **openshift-master-0**
role		ベアメタルノードのロール。**master** (コントロールプレーンノード) または **worker** (コンピュートノード) のいずれか。
bmc		ベースボード管理コントローラーの接続詳細。詳細は、BMC アドレス指定のセクションを参照してください。
bootMACAddress		ホストがプロビジョニングネットワークに使用する NIC の MAC アドレス。Ironic は、**bootMACAddress** 設定を使用して IP アドレスを取得します。次に、ホストにバインドします。 **注記** プロビジョニングネットワークを無効にした場合は、ホストから有効な MAC アドレスを提供する必要があります。
networkConfig		このオプションのパラメーターを設定して、ホストのネットワークインターフェイスを設定します。詳細については、オプション: ホストネットワークインターフェイスの設定を参照してください。

16.3.11.3. BMC アドレス指定

ほとんどのベンダーは、Intelligent Platform Management Interface(IPMI) でベースボード管理コントローラー (BMC) アドレスに対応しています。IPMI は通信を暗号化しません。これは、セキュリティーが保護された管理ネットワークまたは専用の管理ネットワークを介したデータセンター内での使用に適しています。ベンダーを確認して、Redfish ネットワークブートをサポートしているかどうかを確認します。Redfish は、コンバージド、ハイブリッド IT および Software Defined Data Center (SDDC) 向けのシンプルでセキュアな管理を行います。Redfish は人による判読が可能、かつ機械対応が可能であり、インターネットや Web サービスの標準を活用して、最新のツールチェーンに情報を直接公開します。ハードウェアが Redfish ネットワークブートに対応していない場合には、IPMI を使用します。

ノードが **Registering** の状態にある間に、インストール時に BMC アドレスを変更できます。ノードが **Registering** 状態から離れる後に BMC アドレスを変更する必要がある場合は、ノードを Ironic から切断し、**BareMetalHost** リソースを編集して、ノードを Ironic に再接続する必要があります。詳細は、**BareMetalHost リソースの編集** セクションを参照してください。

IPMI

IPMI を使用するホストは **ipmi://<out-of-band-ip>:<port>** アドレス形式を使用します。これは、指定がない場合はポート **623** にデフォルトで設定されます。以下の例は、**install-config.yaml** ファイル内の IPMI 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
```

重要

BMC アドレス指定に IPMI を使用して PXE ブートする場合は、**provisioning** ネットワークが必要です。**provisioning** ネットワークなしでは、PXE ブートホストを行うことはできません。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。詳細は、BMC addressing for HPE iLO セクションの Redfish virtual media for HPE iLO、または BMC addressing for Dell iDRAC セクションの Redfish virtual media for Dell iDRAC セクションを参照してください。

Redfish ネットワークブート

Redfish を有効にするには、**redfish://** または **redfish+http://** を使用して TLS を無効にします。インストーラーには、ホスト名または IP アドレスとシステム ID へのパスの両方が必要です。以下の例は、**install-config.yaml** ファイル内の RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

関連情報

- BareMetalHost リソースの編集

16.3.11.4. Redfish API のサポートの確認

Redfish API を使用してインストールする場合、インストールプログラムは、ベアメタルでインストーラーでプロビジョニングされるインフラストラクチャーを使用する場合に、ベースボード管理コントローラー(BMC)上の Redfish エンドポイントを複数呼び出します。Redfish を使用する場合は、インストール前に BMC がすべての Redfish API をサポートしていることを確認してください。

手順

1. 次のコマンドを実行して、BMC の IP アドレスまたはホスト名を設定します。

```
$ export SERVER=<ip_address> 1
```

1 &lt **;ip_address&gt**; を BMC の IP アドレスまたはホスト名に置き換えます。

2. 次のコマンドを実行して、システムの ID を設定します。

```
$ export SystemID=<system_id> 1
```

1 &lt **;system_id>** をシステム ID に置き換えます。たとえば、**System.Embedded.1** または **1** です。詳細は、以下のベンダー固有の BMC セクションを参照してください。

redfish API のリスト

1. 次のコマンドを実行して、**電源** 投入サポートを確認します。

```
$ curl -u $USER:$PASS -X POST -H'Content-Type: application/json' -H'Accept:
application/json' -d '{"ResetType": "On"}'
https://$SERVER/redfish/v1/Systems/$SystemID/Actions/ComputerSystem.Reset
```

2. 次のコマンドを実行して、**電源オフ** サポートを確認します。

```
$ curl -u $USER:$PASS -X POST -H'Content-Type: application/json' -H'Accept:
application/json' -d '{"ResetType": "ForceOff"}'
https://$SERVER/redfish/v1/Systems/$SystemID/Actions/ComputerSystem.Reset
```

3. 以下のコマンドを実行して、**pxe** を使用する一時的なブート実装を確認します。

```
$ curl -u $USER:$PASS -X PATCH -H "Content-Type: application/json"
https://$Server/redfish/v1/Systems/$SystemID/ -d '{"Boot": {"BootSourceOverrideTarget":
"pxe", "BootSourceOverrideEnabled": "Once"}}
```

4. 次のコマンドを実行して、**Legacy** または **UEFI** を使用する BIOS ブートモードの設定のステータスを確認します。

```
$ curl -u $USER:$PASS -X PATCH -H "Content-Type: application/json"
https://$Server/redfish/v1/Systems/$SystemID/ -d '{"Boot":
{"BootSourceOverrideMode":"UEFI"}}
```

-

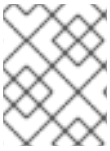
Redfish 仮想メディア API のリスト：

1. 次のコマンドを実行して、**cd** または **dvd** を使用する一時ブートデバイスを設定する機能を確認します。

```
$ curl -u $USER:$PASS -X PATCH -H "Content-Type: application/json"
https://$Server/redfish/v1/Systems/$SystemID/ -d '{"Boot": {"BootSourceOverrideTarget":
"cd", "BootSourceOverrideEnabled": "Once"}}'
```

2. 次のコマンドを実行して、仮想メディアのマウント機能を確認します。

```
$ curl -u $USER:$PASS -X PATCH -H "Content-Type: application/json" -H "If-Match: *"
https://$Server/redfish/v1/Managers/$ManagerID/VirtualMedia/$VmediaId -d '{"Image":
"https://example.com/test.iso", "TransferProtocolType": "HTTPS", "UserName": "",
"Password":""}'
```



注記

redfish API の **PowerOn** および **PowerOff** コマンドは、redfish-virtualmedia API と同じです。

重要

TransferProtocolTypes でサポートされているパラメータータイプは、**HTTPS** と **HTTP** のみです。

16.3.11.5. Dell iDRAC の BMC アドレス指定

それぞれの **bmc** エントリーの **address** フィールドは、URL スキーム内のコントローラーのタイプやネットワーク上のその場所を含む、OpenShift Container Platform クラスターノードに接続する URL です。

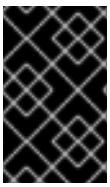
```
platform:
  baremetal:
    hosts:
      - name: <hostname>
        role: <master | worker>
        bmc:
          address: <address> 1
          username: <user>
          password: <password>
```

1 **address** 設定はプロトコルを指定します。

Dell ハードウェアの場合、Red Hat は統合 Dell Remote Access Controller (iDRAC) 仮想メディア、Redfish ネットワークブート、および IPMI をサポートします。

Dell iDRAC の BMC アドレス形式

プロトコル	アドレスのフォーマット
iDRAC 仮想メディア	**idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1**
Redfish ネットワークブート	**redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1**
IPMI	**ipmi://<out-of-band-ip>**



重要

idrac-virtualmedia を Redfish 仮想メディアのプロトコルとして使用します。**redfish-virtualmedia** は Dell ハードウェアでは機能しません。Dell の **idrac-virtualmedia** は、Dell の OEM 拡張機能が含まれる Redfish 標準を使用します。

詳細は、以下のセクションを参照してください。

Dell iDRAC の Redfish 仮想メディア
Dell サーバーの RedFish 仮想メディアについては、 **address** 設定で **idrac-virtualmedia://** を使用します。**redfish-virtualmedia://** を使用しても機能しません。

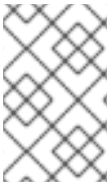
注記

idrac-virtualmedia:// を Redfish 仮想メディアのプロトコルとして使用します。**redfish-virtualmedia://** の使用は、 **idrac-virtualmedia://** プロトコルが **idrac** ハードウェアタイプおよび Ironic の Redfish プロトコルに対応しているため、Dell ハードウェアでは機能しません。Dell の **idrac-virtualmedia://** プロトコルは、Dell の OEM 拡張機能が含まれる Redfish 標準を使用します。Ironic は、WSMAN プロトコルのある **idrac** タイプもサポートします。したがって、Dell ハードウェア上の仮想メディアで Redfish を使用する際に予期しない動作を回避するために、**idrac-virtualmedia://** を指定する必要があります。

以下の例は、**install-config.yaml** ファイル内で iDRAC 仮想メディアを使用する方法を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。



注記

OpenShift Container Platform クラスターノードについて、iDRAC コンソールで **AutoAttach** が有効にされていることを確認します。メニューパスは **Configuration → Virtual Media → Attach Mode → AutoAttach** です。

以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

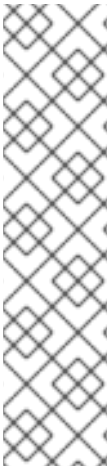
iDRAC の Redfish ネットワークブート

RedFish を有効にするには、**redfish://** または **redfish+http://** を使用してトランスポート層セキュリティー (TLS) を無効にします。インストーラーには、ホスト名または IP アドレスとシステム ID へのパスの両方が必要です。以下の例は、**install-config.yaml** ファイル内の RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```



注記

ファームウェアバージョン **04.40.00.00** を使用する Dell iDRAC 9 と、ベアメタルデプロイメントでインストーラーでプロビジョニングされるインストール用の **5.xx** シリーズを含むすべてのリリースには既知の問題があります。Virtual Console プラグインは、HTML5 の拡張バージョンである eHTML5 にデフォルト設定されているため、**InsertVirtualMedia** ワークフローで問題が発生します。この問題を回避するには、HTML5 を使用するようにプラグインを設定します。メニューパスは以下の通りです。**Configuration → Virtual console → Plug-in Type → HTML5**

OpenShift Container Platform クラスターノードについて、iDRAC コンソールで **AutoAttach** が有効にされていることを確認します。メニューパスは以下のようになります。**Configuration → Virtual Media → Attach Mode → AutoAttach**

16.3.11.6. HPE iLO の BMC アドレス指定

それぞれの **bmc** エントリーの **address** フィールドは、URL スキーム内のコントローラーのタイプやネットワーク上のその場所を含む、OpenShift Container Platform クラスターノードに接続する URL です。

```
platform:
  baremetal:
    hosts:
      - name: <hostname>
        role: <master | worker>
        bmc:
          address: <address> 1
          username: <user>
          password: <password>
```

1 **address** 設定はプロトコルを指定します。

HPE integrated Lights Out (iLO) の場合、Red Hat は Redfish 仮想メディア、Redfish ネットワークブート、および IPMI をサポートします。

表16.8 HPE iLO の BMC アドレス形式

プロトコル	アドレスのフォーマット
Redfish 仮想メディア	**redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1**
Redfish ネットワークブート	**redfish://<out-of-band-ip>/redfish/v1/Systems/1**
IPMI	**ipmi://<out-of-band-ip>**

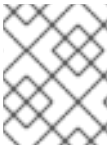
詳細は、以下のセクションを参照してください。

HPE iLO の Redfish 仮想メディア
HPE サーバーについて RedFish Virtual Media を有効にするには、**address** 設定で **redfish-virtualmedia://** を使用します。以下の例は、**install-config.yaml** ファイル内で Redfish 仮想メディアを使用する方法を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```



注記

Ironic は、仮想メディアで iLO4 をサポートしないので、Redfish 仮想メディアは、iLO4 を実行する第 9 世代のシステムではサポートされません。

HPE iLO の Redfish ネットワークブート

Redfish を有効にするには、**redfish://** または **redfish+http://** を使用して TLS を無効にします。インストーラーには、ホスト名または IP アドレスとシステム ID へのパスの両方が必要です。以下の例は、**install-config.yaml** ファイル内の RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
```

```
    - name: openshift-master-0
      role: master
      bmc:
        address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
        username: <user>
        password: <password>
        disableCertificateVerification: True
```

16.3.11.7. Fujitsu iRMC の BMC アドレス指定

それぞれの **bmc** エントリーの **address** フィールドは、URL スキーム内のコントローラーのタイプやネットワーク上のその場所を含む、OpenShift Container Platform クラスターノードに接続する URL です。

```
platform:
  baremetal:
    hosts:
      - name: <hostname>
        role: <master | worker>
        bmc:
          address: <address> 1
          username: <user>
          password: <password>
```

1 **address** 設定はプロトコルを指定します。

Fujitsu ハードウェアの場合、Red Hat は、統合 Remote Management Controller (iRMC) および IPMI をサポートします。

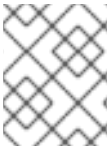
表16.9 Fujitsu iRMC の BMC アドレス形式

プロトコル	アドレスのフォーマット
iRMC	**irmc://<out-of-band-ip>**
IPMI	**ipmi://<out-of-band-ip>**

iRMC

Fujitsu ノードは **irmc://<out-of-band-ip>** を使用し、デフォルトではポート **443** に設定されます。以下の例は、**install-config.yaml** ファイル内の iRMC 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: irmc://<out-of-band-ip>
          username: <user>
          password: <password>
```



注記

現在 Fujitsu は、ベアメタルへのインストーラーでプロビジョニングされるインストール用に iRMC S5 ファームウェアバージョン 3.05P 以降をサポートしています。

16.3.11.8. ルートデバイスのヒント

rootDeviceHints パラメーターは、インストーラーが Red Hat Enterprise Linux CoreOS (RHCOS) イメージを特定のデバイスにプロビジョニングできるようにします。インストーラーは、検出順にデバイスを検査し、検出された値をヒントの値と比較します。インストーラーは、ヒント値に一致する最初に検出されたデバイスを使用します。この設定は複数のヒントを組み合わせることができますが、デバイスは、インストーラーがこれを選択できるようにすべてのヒントに一致する必要があります。

表16.10 サブフィールド

サブフィールド	説明
deviceName	**/dev/vda** や **/dev/disk/by-path/** などの Linux デバイス名を含む文字列。ストレージの場所への **/dev/disk/by-path/<device_path>** リンクを使用することを推奨します。ヒントは、実際の値と完全に一致する必要があります。
hctl	**0:0:0:0** などの SCSI バスアドレスを含む文字列。ヒントは、実際の値と完全に一致する必要があります。
model	ベンダー固有のデバイス識別子を含む文字列。ヒントは、実際の値のサブ文字列になります。
vendor	デバイスのベンダーまたは製造元の名前が含まれる文字列。ヒントは、実際の値のサブ文字列になります。
serialNumber	デバイスのシリアル番号を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
minSizeGigabytes	デバイスの最小サイズ (ギガバイト単位) を表す整数。
wwn	一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
wwnWithExtension	ベンダー拡張が追加された一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
wwnVendorExtension	一意のベンダーストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
rotational	デバイスがローテーションするディスクである (true) か、そうでないか (false) を示すブール値。

使用例

```
    - name: master-0
      role: master
      bmc:
        address: ipmi://10.10.0.3:6203
        username: admin
        password: redhat
      bootMACAddress: de:ad:be:ef:00:40
      rootDeviceHints:
        deviceName: "/dev/sda"
```

16.3.11.9. オプション： プロキシー設定の設定

プロキシーを使用して OpenShift Container Platform クラスターをデプロイするには、 **install-config.yaml** ファイルに以下の変更を加えます。

```
apiVersion: v1
baseDomain: <domain>
proxy:
  httpProxy: http://USERNAME:PASSWORD@proxy.example.com:PORT
  httpsProxy: https://USERNAME:PASSWORD@proxy.example.com:PORT
  noProxy: <WILDCARD_OF_DOMAIN>,<PROVISIONING_NETWORK/CIDR>,
<BMC_ADDRESS_RANGE/CIDR>
```

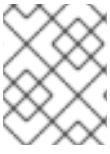
以下は、値を含む **noProxy** の例です。

```
noProxy: .example.com,172.22.0.0/24,10.10.0.0/24
```

プロキシーを有効な状態にして、対応するキー/値のペアでプロキシーの適切な値を設定します。

主な留意事項:

- プロキシーに HTTPS プロキシーがない場合、 **httpsProxy** の値を **https://** から **http://** に変更します。
- provisioning ネットワークを使用する場合、これを **noProxy** 設定に含めます。そうしない場合、インストーラーは失敗します。
- プロビジョナーノード内の環境変数としてすべてのプロキシー設定を設定します。たとえば、**HTTP_PROXY**、**HTTPS_PROXY**、および **NO_PROXY** が含まれます。



注記

IPv6 でプロビジョニングする場合、**noProxy** 設定で CIDR アドレスブロックを定義することはできません。各アドレスを個別に定義する必要があります。

16.3.11.10. オプション: プロビジョニングネットワークを使用しないデプロイ

provisioning ネットワークなしに OpenShift Container Platformmake をデプロイするには、 **install-config.yaml** ファイルに以下の変更を加えます。

```
platform:
  baremetal:
```

```
apiVIPs:
  - <api_VIP>
ingressVIPs:
  - <ingress_VIP>
provisioningNetwork: "Disabled" (1)
```

(1) **provisioningNetwork** 設定を追加して、必要な場合はこれを **Disabled** に設定します。

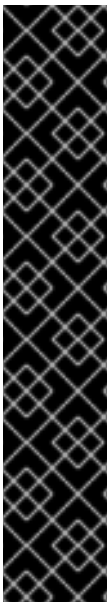
重要

PXE ブートには **provisioning** ネットワークが必要です。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。詳細は、BMC addressing for HPE iLO セクションの Redfish virtual media for HPE iLO、または BMC addressing for Dell iDRAC セクションの Redfish virtual media for Dell iDRAC セクションを参照してください。

16.3.11.11. オプション: デュアルスタックネットワークを使用したデプロイ

OpenShift Container Platform クラスターのデュアルスタックネットワークでは、クラスターノードの IPv4 および IPv6 アドレスエンドポイントを設定できます。クラスターノードの IPv4 および IPv6 アドレスエンドポイントを設定するには、**install-config.yaml** ファイルで **machineNetwork**、**clusterNetwork**、および **serviceNetwork** 設定を編集します。それぞれの設定には、それぞれ 2 つの CIDR エントリーが必要です。プライマリーアドレスファミリーとして IPv4 ファミリーを持つクラスターの場合は、最初に IPv4 設定を指定します。プライマリーアドレスファミリーとして IPv6 ファミリーを持つクラスターの場合は、最初に IPv6 設定を指定します。

```
machineNetwork:
- cidr: {{ extcidrnet }}
- cidr: {{ extcidrnet6 }}
clusterNetwork:
- cidr: 10.128.0.0/14
  hostPrefix: 23
- cidr: fd02::/48
  hostPrefix: 64
serviceNetwork:
- 172.30.0.0/16
- fd03::/112
```



重要

install-config.yaml ファイルの **networkConfig** セクションで NMState 設定を指定した場合は、クラスターをデュアルスタックネットワークにデプロイできない問題を解決するために、**interfaces.wait-ip: ipv4+ipv6** を NMState YAML ファイルに追加してください。

wait-ip パラメーターを含む NMState YAML 設定ファイルの例

```
networkConfig:
  nmstate:
    interfaces:
    - name: <interface_name>
# ...
      wait-ip: ipv4+ipv6
# ...
```

IPv4 および IPv6 アドレスを使用するアプリケーションのクラスターへのインターフェイスを提供するには、Ingress VIP および API VIP サービスの IPv4 および IPv6 仮想 IP (VIP) アドレスエンドポイントを設定します。IPv4 および IPv6 アドレスエンドポイントを設定するには、**install-config.yaml** ファイルで **apiVIPs** および **ingressVIPs** 設定を編集します。**apiVIPs** および **ingressVIPs** 設定では、リスト形式を使用します。リストの順序は、各サービスのプライマリーおよびセカンダリー VIP アドレスを示しています。

```
platform:
  baremetal:
    apiVIPs:
      - <api_ipv4>
      - <api_ipv6>
    ingressVIPs:
      - <wildcard_ipv4>
      - <wildcard_ipv6>
```

注記

デュアルスタックネットワーク設定のクラスターの場合、IPv4 アドレスと IPv6 アドレスの両方を同じインターフェイスに割り当てる必要があります。

16.3.11.12. オプション: ホストネットワークインターフェイスの設定

インストールの前に、**install-config.yaml** ファイルで **networkConfig** 設定を設定し、NMState を使用してホストネットワークインターフェイスを設定できます。

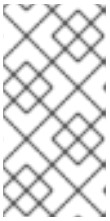
この機能の最も一般的な使用例は、ベアメタルネットワークで静的 IP アドレスを指定することですが、ストレージネットワークなどの他のネットワークを設定することもできます。この機能は、VLAN、VXLAN、ブリッジ、ボンド、ルート、MTU、DNS リゾルバー設定など、他の NMState 機能をサポートします。

前提条件

- 静的 IP アドレスを持つ各ノードの有効なホスト名で **PTR** DNS レコードを設定する。
- NMState CLI (**nmstate**) をインストールする。

手順

1. オプション: インストーラーは NMState YAML 構文をチェックしないため、**install-config.yaml** ファイルに含める前に **nmstatectl gc** で NMState 構文をテストすることを検討してください。



注記

YAML 構文にエラーがあると、ネットワーク設定の適用に失敗する可能性があります。さらに、検証済みの YAML 構文を維持すると、デプロイ後に Kubernetes NMState を使用して変更を適用する場合、またはクラスターを拡張する場合に役立ちます。

   a. NMState YAML ファイルを作成します。

```
interfaces:
- name: <nic1_name> (1)
  type: ethernet
  state: up
  ipv4:
    address:
    - ip: <ip_address> (2)
      prefix-length: 24
    enabled: true
dns-resolver:
  config:
    server:
    - <dns_ip_address> (3)
routes:
  config:
  - destination: 0.0.0.0/0
    next-hop-address: <next_hop_ip_address> (4)
    next-hop-interface: <next_hop_nic1_name> (5)
```

(1) (2) (3) (4) (5) **<nic1_name>**、**<ip_address>**、**<dns_ip_address>**、**<next_hop_ip_address>**、および **<next_hop_nic1_name>** を適切な値に置き換えます。

   b. 次のコマンドを実行して、設定ファイルをテストします。

```
$ nmstatectl gc <nmstate_yaml_file>
```

<nmstate_yaml_file> を設定ファイル名に置き換えます。

2. **install-config.yaml** ファイル内のホストに NMState 設定を追加して、 **networkConfig** 設定を使用します。

```
hosts:
  - name: openshift-master-0
    role: master
    bmc:
      address: redfish+http://<out_of_band_ip>/redfish/v1/Systems/
      username: <user>
```

```
    password: <password>
    disableCertificateVerification: null
  bootMACAddress: <NIC1_mac_address>
  bootMode: UEFI
  rootDeviceHints:
    deviceName: "/dev/sda"
  networkConfig: 1
    interfaces:
    - name: <nic1_name> 2
      type: ethernet
      state: up
      ipv4:
        address:
        - ip: <ip_address> 3
          prefix-length: 24
        enabled: true
    dns-resolver:
      config:
        server:
        - <dns_ip_address> 4
    routes:
      config:
      - destination: 0.0.0.0/0
        next-hop-address: <next_hop_ip_address> 5
        next-hop-interface: <next_hop_nic1_name> 6
```

1 NMState YAML 構文を追加して、ホストインターフェイスを設定します。

2 3 4 5 6 **<nic1_name>**、**<ip_address>**、**<dns_ip_address>**、**<next_hop_ip_address>**、および **<next_hop_nic1_name>** を適切な値に置き換えます。

重要

クラスターをデプロイした後、**install-config.yaml** ファイルの **networkConfig** 設定を変更して、ホストネットワークインターフェイスを変更することはできません。Kubernetes NMState Operator を使用して、デプロイ後にホストネットワークインターフェイスに変更を加えます。

16.3.11.13. サブネット用のホストネットワークインターフェイスの設定

エッジコンピューティングのシナリオでは、コンピュートノードをエッジの近くに配置することが有益な場合があります。サブネット内のリモートノードを見つけるために、コントロールプレーンサブネットやローカルコンピュートノードに使用したものとは異なるネットワークセグメントまたはサブネットをリモートノードに使用できます。エッジコンピューティングシナリオ用にサブネットを設定することで、エッジのレイテンシーが短縮され、拡張性が向上します。

重要

デフォルトのロードバランサー **OpenShiftManagedDefault** を使用してリモートノードを OpenShift Container Platform クラスターに追加する場合は、すべてのコントロールプレーンノードが同じサブネット内で実行されている必要があります。複数のサブネットを使用する場合、マニフェストを使用して、コントロールプレーンノード上で実行されるように Ingress VIP を設定することもできます。詳細は、「コントロールプレーンで実行するネットワークコンポーネントの設定」を参照してください。

「サブネット間の通信を確立する」セクションの説明どおりにリモートノードに異なるネットワークセグメントまたはサブネットを確立した場合、ワーカーが静的 IP アドレス、ボンディング、またはその他の高度なネットワークを使用している場合は、**machineNetwork** 設定でサブネットを指定する必要があります。各リモートノードの **networkConfig** パラメーターでノード IP アドレスを設定する場合、静的 IP アドレスを使用する際に、コントロールプレーンノードを含むサブネットのゲートウェイと DNS サーバーも指定する必要があります。これにより、リモートノードがコントロールプレーンノードを含むサブネットに到達し、コントロールプレーンからネットワークトラフィックを受信できるようになります。

注記

複数のサブネットを持つクラスターをデプロイするには、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディアを使用する必要があります。リモートノードがローカルプロビジョニングネットワークにアクセスできないためです。

手順

1. 静的 IP アドレスを使用する場合は、**install-config.yaml** ファイルの **machineNetwork** にサブネットを追加します。

```
networking:
  machineNetwork:
  - cidr: 10.0.0.0/24
  - cidr: 192.168.0.0/24
  networkType: OVNKubernetes
```

2. 静的 IP アドレスまたはボンディングなどの高度なネットワークを使用する場合は、NMState 構文を使用して、各エッジコンピュートノードの **networkConfig** パラメーターにゲートウェイと DNS 設定を追加します。

```
networkConfig:
  interfaces:
  - name: <interface_name> 1
    type: ethernet
    state: up
    ipv4:
      enabled: true
      dhcp: false
      address:
      - ip: <node_ip> 2
        prefix-length: 24
      gateway: <gateway_ip> 3
      dns-resolver:
```

```
    config:
      server:
      - <dns_ip> 4
```

1. **<interface_name>** をインターフェイス名に置き換えます。
2. **<node_ip>** をノードの IP アドレスに置き換えます。
3. **<gateway_ip>** をゲートウェイの IP アドレスに置き換えます。
4. **<dns_ip>** を DNS サーバーの IP アドレスに置き換えます。

16.3.11.14. オプション: デュアルスタックネットワーク内の SLAAC のアドレス生成モードを設定する

Stateless Address AutoConfiguration (SLAAC) を使用するデュアルスタッククラスターの場合は、**ipv6.addr-gen-mode** ネットワーク設定のグローバル値を指定する必要があります。NMState を使用してこの値を設定し、RAM ディスクとクラスター設定ファイルを設定できます。これらの場所で一貫した **ipv6.addr-gen-mode** を設定しないと、クラスター内の CSR リソースと **BareMetalHost** リソース間で IPv6 アドレスの不一致が発生する可能性があります。

前提条件

- NMState CLI (**nmstate**) をインストールする。

手順

1. オプション: インストールプログラムは NMState YAML 構文をチェックしないため、**install-config.yaml** ファイルに含める前に **nmstatectl gc** コマンドを使用して NMState YAML 構文をテストすることを検討してください。

   a. NMState YAML ファイルを作成します。

   ```
   interfaces:
   - name: eth0
     ipv6:
       addr-gen-mode: <address_mode> 1
   ```

   1. **<address_mode>** を、クラスター内の IPv6 アドレスに必要なアドレス生成モードのタイプに置き換えます。有効な値は、**eui64**、**steady-privacy**、または **random** です。

   b. 次のコマンドを実行して、設定ファイルをテストします。

   ```
   $ nmstatectl gc <nmstate_yaml_file> 1
   ```

   1. **<nmstate_yaml_file>** を、テスト設定ファイルの名前に置き換えます。

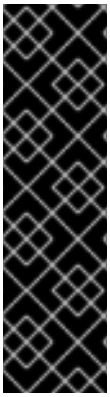
2. NMState 設定を、install-config.yaml ファイル内の **hosts.networkConfig** セクションに追加します。

```
    hosts:
```

```
 - name: openshift-master-0
   role: master
   bmc:
     address: redfish+http://<out_of_band_ip>/redfish/v1/Systems/
     username: <user>
     password: <password>
     disableCertificateVerification: null
   bootMACAddress: <NIC1_mac_address>
   bootMode: UEFI
   rootDeviceHints:
     deviceName: "/dev/sda"
   networkConfig:
     interfaces:
     - name: eth0
       ipv6:
         addr-gen-mode: <address_mode> 1
...
```

1 **<address_mode>** を、クラスター内の IPv6 アドレスに必要なアドレス生成モードのタイプに置き換えます。有効な値は、**eui64**、**steady-privacy**、または **random** です。

16.3.11.15. オプション: デュアルポート NIC 用のホストネットワークインターフェイスの設定



重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

インストールの前に、**install-config.yaml** ファイルで **networkConfig** 設定を行って、デュアルポート NIC をサポートするように、NMState を使用して、ホストネットワークインターフェイスを設定できます。

前提条件

- 静的 IP アドレスを持つ各ノードの有効なホスト名で **PTR** DNS レコードを設定する。
- NMState CLI (**nmstate**) をインストールする。

注記

YAML 構文にエラーがあると、ネットワーク設定の適用に失敗する可能性があります。さらに、検証済みの YAML 構文を維持すると、デプロイ後に Kubernetes NMState を使用して変更を適用する場合、またはクラスターを拡張する場合に役立ちます。

手順

1. **install-config.yaml** ファイル内のホストの **networkConfig** フィールドに NMState 設定を追加します。

```
hosts:
  - name: worker-0
    role: worker
    bmc:
      address: redfish+http://<out_of_band_ip>/redfish/v1/Systems/
      username: <user>
      password: <password>
      disableCertificateVerification: false
    bootMACAddress: <NIC1_mac_address>
    bootMode: UEFI
    networkConfig: (1)
      interfaces: (2)
       - name: eno1 (3)
         type: ethernet (4)
         state: up
         mac-address: 0c:42:a1:55:f3:06
         ipv4:
           enabled: true
           dhcp: false (5)
         ethernet:
           sr-iov:
             total-vfs: 2 (6)
         ipv6:
           enabled: false
           dhcp: false
       - name: sriov:eno1:0
         type: ethernet
         state: up (7)
         ipv4:
           enabled: false (8)
         ipv6:
           enabled: false
       - name: sriov:eno1:1
         type: ethernet
         state: down
       - name: eno2
         type: ethernet
         state: up
         mac-address: 0c:42:a1:55:f3:07
         ipv4:
           enabled: true
         ethernet:
           sr-iov:
             total-vfs: 2
         ipv6:
           enabled: false
       - name: sriov:eno2:0
         type: ethernet
         state: up
         ipv4:
           enabled: false
         ipv6:
```

```
      enabled: false
    - name: sriov:eno2:1
      type: ethernet
      state: down
    - name: bond0
      type: bond
      state: up
      min-tx-rate: 100 9
      max-tx-rate: 200 10
      link-aggregation:
        mode: active-backup 11
        options:
          primary: sriov:eno1:0 12
        port:
          - sriov:eno1:0
          - sriov:eno2:0
      ipv4:
        address:
          - ip: 10.19.16.57 13
            prefix-length: 23
        dhcp: false
        enabled: true
      ipv6:
        enabled: false
      dns-resolver:
        config:
          server:
            - 10.11.5.160
            - 10.2.70.215
      routes:
        config:
          - destination: 0.0.0.0/0
            next-hop-address: 10.19.17.254
            next-hop-interface: bond0 14
            table-id: 254
```

1. **networkConfig** フィールドには、ホストのネットワーク設定に関する情報が含まれており、サブフィールドには、**interfaces**、**dns-resolver**、および **Routes** が含まれます。
2. **interfaces** フィールドは、ホスト用に定義されたネットワークインターフェイスの配列です。
3. インターフェイスの名前。
4. インターフェイスのタイプ。この例では、イーサネットインターフェイスを作成します。
5. 厳密に必要ではない場合、物理機能 (PF) の DHCP を無効にするには、これを false に設定します。
6. インスタンス化する SR-IOV 仮想機能 (VF) の数に設定します。
7. これを **up** に設定します。
8. ボンドに接続された VF の IPv4 アドレス指定を無効にするには、これを **false** に設定します。

9 VF の最小伝送速度 (Mbps) を設定します。このサンプル値は、100 Mbps のレートを設定します。

- この値は、最大伝送レート以下である必要があります。
- Intel NIC は **min-tx-rate** パラメーターをサポートしていません。詳細については、BZ#1772847 を参照してください。

10 VF の最大伝送速度 (Mbps) を設定します。このサンプル値は、200 Mbps のレートを設定します。

11 目的のボンディングモードを設定します。

12 ボンディングインターフェイスの優先ポートを設定します。プライマリーデバイスは、最初に使用されるボンディングインターフェイスであり、障害が発生しないかぎり、破棄されません。この設定が特に役立つのは、ボンディングインターフェイスの NIC の 1 つが高速なため、大規模な負荷に対応できる場合です。この設定は、ボンディングインターフェイスが active-backup モード (モード 1) および balance-tlb (モード 5) の場合のみに有効です。

13 ボンドインターフェイスの静的 IP アドレスを設定します。これはノードの IP アドレスです。

14 デフォルトルートのゲートウェイとして **bond0** を設定します。

重要

クラスターをデプロイした後、**install-config.yaml** ファイルの **networkConfig** 設定を変更して、ホストネットワークインターフェイスを変更することはできません。Kubernetes NMState Operator を使用して、デプロイ後にホストネットワークインターフェイスに変更を加えます。

関連情報

- ネットワークボンディングの設定

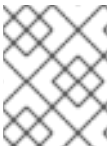
16.3.11.16. 複数のクラスターノードの設定

同じ設定で OpenShift Container Platform クラスターノードを同時に設定できます。複数のクラスターノードを設定すると、各ノードの冗長な情報が **install-config.yaml** ファイルに追加されることを回避できます。このファイルには、クラスター内の複数のノードに同一の設定を適用するための特定のパラメーターが含まれています。

コンピュートノードは、コントローラーノードとは別に設定されます。ただし、両方のノードタイプの設定では、**install-config.yaml** ファイルで強調表示されているパラメーターを使用して、マルチノード設定を有効にします。次の例に示すように、**networkConfig** パラメーターを **BOND** に設定します。

```
hosts:
- name: ostest-master-0
 [...]
 networkConfig: &BOND
   interfaces:
   - name: bond0
     type: bond
```

```
      state: up
      ipv4:
        dhcp: true
        enabled: true
      link-aggregation:
        mode: active-backup
        port:
        - enp2s0
        - enp3s0
- name: ostest-master-1
 [...]
 networkConfig: *BOND
- name: ostest-master-2
 [...]
 networkConfig: *BOND
```



注記

複数のクラスターノードの設定は、インストーラーによってプロビジョニングされたインフラストラクチャーでの初期デプロイメントでのみ使用できます。

16.3.11.17. オプション: マネージドセキュアブートの設定

redfish、**redfish-virtualmedia**、**idrac-virtualmedia** などの Redfish BMC アドレス指定を使用して、インストーラーでプロビジョニングされたクラスターをデプロイする際に管理対象 Secure Boot を有効にすることができます。マネージド Secure Boot を有効にするには、**bootMode** 設定を各ノードに追加します。

例

```
hosts:
  - name: openshift-master-0
    role: master
    bmc:
      address: redfish://<out_of_band_ip> (1)
      username: <username>
      password: <password>
    bootMACAddress: <NIC1_mac_address>
    rootDeviceHints:
     deviceName: "/dev/sda"
    bootMode: UEFISecureBoot (2)
```

(1) **bmc.address** 設定は、**redfish**、**redfish-virtualmedia**、または **idrac-virtualmedia** をプロトコルとして使用することを確認します。詳細は、BMC addressing for HPE iLO または BMC addressing for Dell iDRAC を参照してください。

(2) **bootMode** 設定は、デフォルトで **UEFI** となります。これを **UEFISecureBoot** に変更して、マネージド Secure Boot を有効にします。

注記

ノードがマネージド Secure Boot をサポートできるようにするには、前提条件のノードの設定を参照してください。ノードがマネージド Secure Boot に対応していない場合には、ノードの設定セクションの手動での Secure Boot のノードの設定を参照してください。Secure Boot を手動で設定するには、Redfish 仮想メディアが必要です。

注記

IPMI は Secure Boot 管理機能を提供しないため、Red Hat では IPMI による Secure Boot のサポートはありません。

16.3.12. マニフェスト設定ファイル

16.3.12.1. OpenShift Container Platform マニフェストの作成

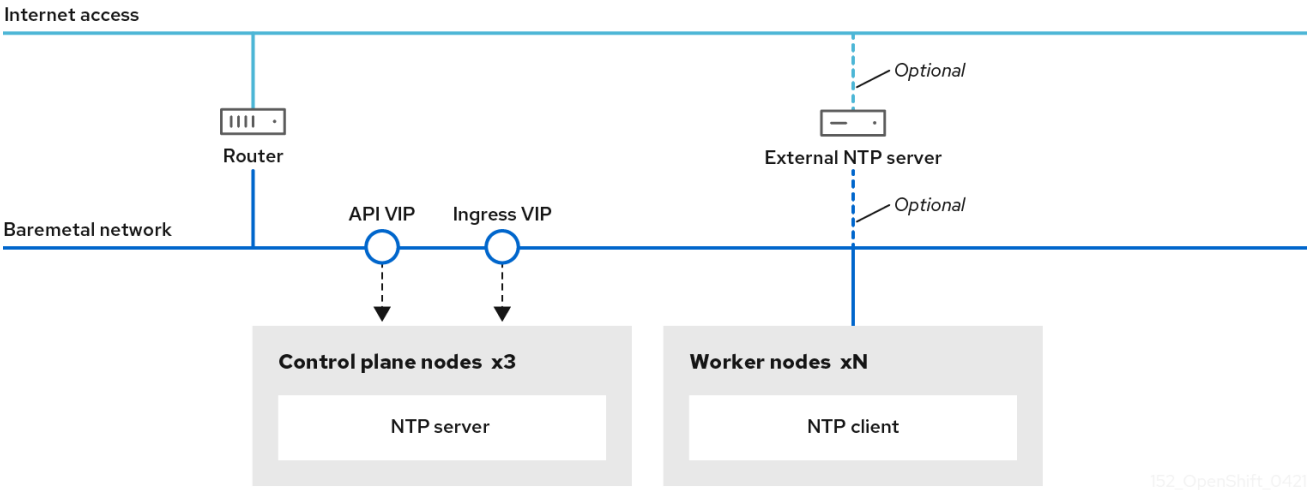
1. OpenShift Container Platform マニフェストを作成します。

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs create manifests
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
WARNING Discarding the OpenShift Manifest that was provided in the target directory
because its dependencies are dirty and it needs to be regenerated
```

16.3.12.2. オプション: 非接続クラスターの NTP 設定

OpenShift Container Platform は、クラスターノードに **chrony** Network Time Protocol (NTP) サービスをインストールします。



OpenShift Container Platform のノードは、適切に実行するために日付と時刻が一致している必要があります。コンピュートノードがコントロールプレーンノード上の NTP サーバーから日付と時刻を取得すると、ルーティング可能なネットワークに接続していないために上位層の NTP サーバーにアクセスできないクラスターのインストールと操作が可能になります。

手順

1. コントロールプレーンノードの **chrony.conf** ファイルのコンテンツを含む Butane 設定 (**99-master-chrony-conf-override.bu**) を作成します。

注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

Butane 設定例

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-master-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # Use public servers from the pool.ntp.org project.
          # Please consider joining the pool (https://www.pool.ntp.org/join.html).

          # The Machine Config Operator manages this file
          server openshift-master-0.<cluster-name>.<domain> iburst 1
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
          keyfile /etc/chrony.keys
          commandkey 1
          generatecommandkey
          noclientlog
          logchange 0.5
          logdir /var/log/chrony

          # Configure the control plane nodes to serve as local NTP servers
          # for all compute nodes, even if they are not in sync with an
          # upstream NTP server.

          # Allow NTP client access from the local network.
          allow all
          # Serve time even if not synchronized to a time source.
          local stratum 3 orphan
```

1 **<cluster-name>** はクラスターの名前に置き換え、**<domain>** は完全修飾ドメイン名に置き換える必要があります。

2. Butane を使用して、コントロールプレーンノードに配信される設定が含まれる **MachineConfig** オブジェクトファイル (**99-master-chrony-conf-override.yaml**) を生成します。

```
$ butane 99-master-chrony-conf-override.bu -o 99-master-chrony-conf-override.yaml
```

3. コントロールプレーンノード上の NTP サーバーを参照するコンピュートノードの **chrony.conf** ファイルの内容を含む Butane 設定 **99-worker-chrony-conf-override.bu** を作成します。

Butane 設定例

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # The Machine Config Operator manages this file.
          server openshift-master-0.<cluster-name>.<domain> iburst (1)
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
          keyfile /etc/chrony.keys
          commandkey 1
          generatecommandkey
          noclientlog
          logchange 0.5
          logdir /var/log/chrony
```

(1) **<cluster-name>** はクラスターの名前に置き換え、**<domain>** は完全修飾ドメイン名に置き換える必要があります。

4. Butane を使用して、ワーカーノードに配信される設定が含まれる **MachineConfig** オブジェクトファイル (**99-worker-chrony-conf-override.yaml**) を生成します。

```
$ butane 99-worker-chrony-conf-override.bu -o 99-worker-chrony-conf-override.yaml
```

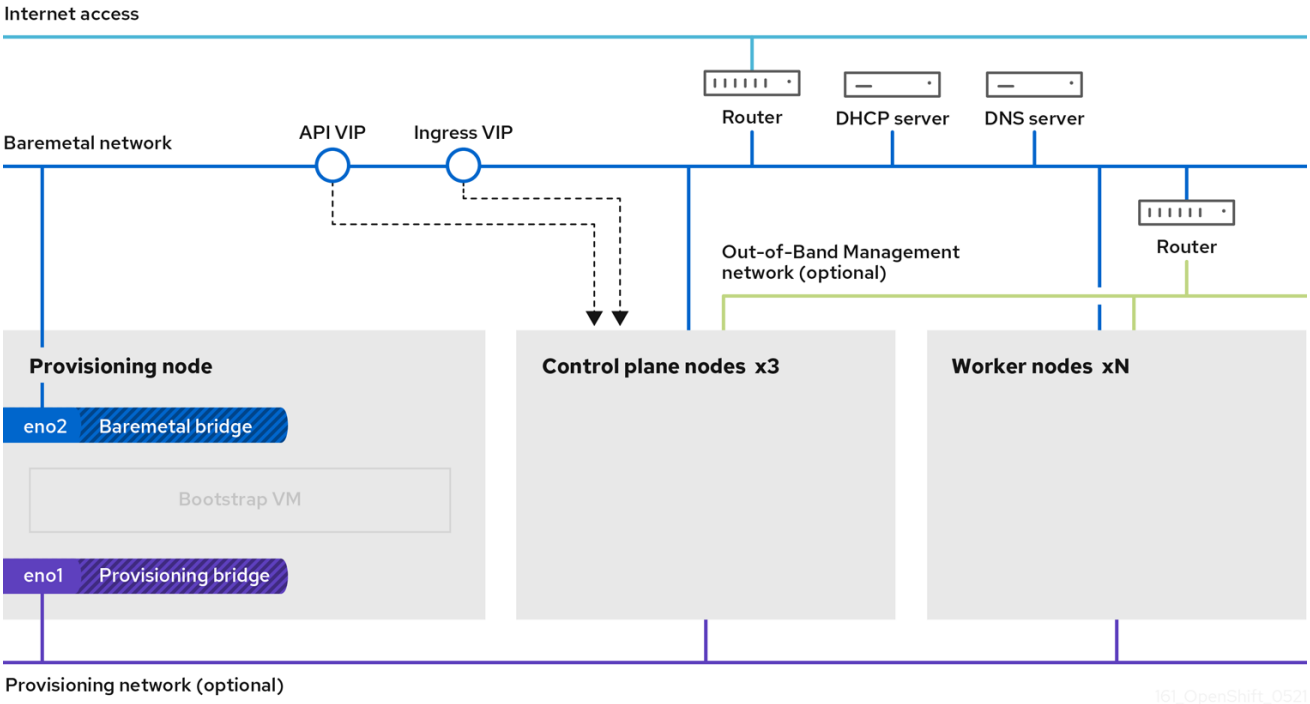
16.3.12.3. コントロールプレーンで実行されるネットワークコンポーネントの設定

ネットワークコンポーネントは、コントロールプレーンノードでのみ実行するように設定できます。デ

フォルトで、OpenShift Container Platform はマシン設定プールの任意のノードが **ingressVIP** 仮想 IP アドレスをホストできるようにします。ただし、環境によっては、コントロールプレーンノードとは別のサブネットにコンピュートノードがデプロイされるため、コントロールプレーンノードで実行するために **ingressVIP** 仮想 IP アドレスを設定する必要があります。

重要

リモートノードを別々のサブネットにデプロイする場合は、コントロールプレーンノード専用に **ingressVIP** 仮想 IP アドレスを配置する必要があります。



手順

1. **install-config.yaml** ファイルを保存するディレクトリーに移動します。

```
$ cd ~/clusterconfigs
```

2. **manifests** サブディレクトリーに切り替えます。

```
$ cd manifests
```

3. **cluster-network-avoid-workers-99-config.yaml** という名前のファイルを作成します。

```
$ touch cluster-network-avoid-workers-99-config.yaml
```

4. エディターで **cluster-network-avoid-workers-99-config.yaml** ファイルを開き、Operator 設定を記述するカスタムリソース (CR) を入力します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 50-worker-fix-ipi-rwn
```

```
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/kubernetes/manifests/keepalived.yaml
          mode: 0644
          contents:
            source: data:,
```

このマニフェストは、 **ingressVIP** 仮想 IP アドレスをコントロールプレーンノードに配置します。また、このマニフェストは、コントロールプレーンノードにのみ以下のプロセスをデプロイします。

- **openshift-ingress-operator**
- **keepalived**

5. **cluster-network-avoid-workers-99-config.yaml** ファイルを保存します。
6. **manifests/cluster-ingress-default-ingresscontroller.yaml** ファイルを作成します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/master: ""
```

7. **manifests** ディレクトリーのバックアップの作成を検討してください。インストーラーは、クラスターの作成時に **manifests/** ディレクトリーを削除します。
8. **cluster-scheduler-02-config.yml** マニフェストを変更し、**mastersSchedulable** フィールドを **true** に設定して、コントロールプレーンノードをスケジュール対象にします。デフォルトでは、コントロールプレーンノードはスケジュール対象ではありません。以下に例を示します。

```
$ sed -i "s;mastersSchedulable: false;mastersSchedulable: true;g"
clusterconfigs/manifests/cluster-scheduler-02-config.yml
```

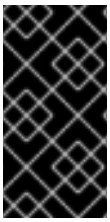
注記

この手順の完了後にコントロールプレーンノードをスケジュールできない場合には、クラスターのデプロイに失敗します。

16.3.12.4. オプション: コンピュートノードへのルーターのデプロイ

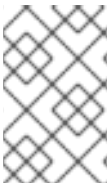
インストール時に、インストールプログラムはルーター Pod をコンピュートノードにデプロイします。デフォルトでは、インストールプログラムは 2 つのルーター Pod をインストールします。デプロ

イされたクラスターが、OpenShift Container Platform クラスター内のサービスに対して予定される外部トラフィック負荷を処理するために追加のルーターを必要とする場合、**yaml** ファイルを作成して適切なルーターレプリカ数を設定できます。



重要

コンピュートノードが 1 つだけのクラスターのデプロイはサポートされていません。ルーターのレプリカを変更すると、1 つのコンピュートノードでデプロイする場合の **degraded** 状態の問題が解決されますが、クラスターで Ingress API の高可用性が失われるため、実稼働環境には適していません。



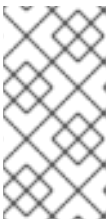
注記

デフォルトでは、インストールプログラムは 2 つのルーターをデプロイします。クラスターにコンピュートノードがない場合、インストールプログラムはデフォルトで 2 つのルーターをコントロールプレーンノードにデプロイします。

手順

1. **router-replicas.yaml** ファイルを作成します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: <num-of-router-pods>
  endpointPublishingStrategy:
    type: HostNetwork
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/worker: ""
```



注記

<num-of-router-pods> を適切な値に置き換えます。1 つのコンピュートノードのみを使用する場合は、**replicas:** を **1** に設定します。3 つ以上のコンピュートノードを使用する場合は、必要に応じて、**replicas:** をデフォルト値 **2** から増やすことができます。

2. **router-replicas.yaml** ファイルを保存し、これを **clusterconfigs/openshift** ディレクトリーにコピーします。

```
$ cp ~/router-replicas.yaml clusterconfigs/openshift/99_router-replicas.yaml
```

16.3.12.5. オプション: BIOS の設定

次の手順では、インストールプロセス中に BIOS を設定します。

手順

1. マニフェストを作成します。
2. ノードに対応する **BareMetalHost** リソースファイルを変更します。

```
$ vim clusterconfigs/openshift/99_openshift-cluster-api_hosts-*.yaml
```

3. BIOS 設定を **BareMetalHost** リソースの **spec** セクションに追加します。

```
spec:
  firmware:
    simultaneousMultithreadingEnabled: true
    sriovEnabled: true
    virtualizationEnabled: true
```

注記

Red Hat は 3 つの BIOS 設定をサポートしています。BMC タイプ**irmc**のサーバーのみがサポートされます。他のタイプのサーバーは現在サポートされていません。

4. クラスターを作成します。

関連情報

- ベアメタルの設定

16.3.12.6. オプション: RAID の設定

次の手順では、インストールプロセス中にベースボード管理コントローラー (BMC) を使用して Redundant Array of Independent Disks (RAID) を構成します。

注記

ノードにハードウェア RAID を設定する場合は、ノードにサポートされている RAID コントローラーがあることを確認してください。OpenShift Container Platform 4.16 は、ソフトウェア RAID をサポートしていません。

表16.11 ベンダーによるハードウェア RAID のサポート

Vendor	BMC とプロトコル	ファームウェアのバージョン	RAID レベル
Fujitsu	iRMC	該当なし	0、1、5、6、10
Dell	iDRAC と Redfish	バージョン 6.10.30.20 以降	0、1、5

手順

1. マニフェストを作成します。

2. ノードに対応する **BareMetalHost** リソースを変更します。

```
$ vim clusterconfigs/openshift/99_openshift-cluster-api_hosts-*.yaml
```

注記

OpenShift Container Platform 4.16 はソフトウェア RAID をサポートしていないため、次の例ではハードウェア RAID 設定を使用します。

a. 特定の RAID 設定を **spec** セクションに追加した場合、これが原因でノードは **preparing** フェーズで元の RAID 設定を削除し、RAID で指定された設定を実行します。以下に例を示します。

```
spec:
  raid:
    hardwareRAIDVolumes:
    - level: "0" (1)
      name: "sda"
      numberOfPhysicalDisks: 1
      rotational: true
      sizeGibibytes: 0
```

(1) **level**は必須フィールドであり、その他はオプションのフィールドです。

b. **spec** セクションに空の RAID 設定を追加した場合、空の設定が原因で、ノードは **preparing** フェーズで元の RAID 設定を削除しますが、新しい設定は実行しません。以下に例を示します。

```
spec:
  raid:
    hardwareRAIDVolumes: []
```

c. **spec** セクションに **raid** フィールドを追加しない場合、元の RAID 設定は削除されず、新しい設定は実行されません。

3. クラスターを作成します。

16.3.12.7. オプション: ノード上のストレージの設定

Machine Config Operator (MCO) によって管理される **MachineConfig** オブジェクトを作成することにより、OpenShift Container Platform ノード上のオペレーティングシステムに変更を加えることができます。

MachineConfig 仕様には、最初の起動時にマシンを設定するための点火設定が含まれています。この設定オブジェクトを使用して、OpenShift Container Platform マシン上で実行されているファイル、systemd サービス、およびその他のオペレーティングシステム機能を変更できます。

手順

ignition config を使用して、ノード上のストレージを設定します。次の **MachineSet** マニフェストの例は、プライマリーノード上のデバイスにパーティションを追加する方法を示しています。この例では、インストール前にマニフェストを適用して、プライマリーノードでサイズが 16 GiB の **recovery** という名前のパーティションを設定します。

1. **Custom-partitions.yaml** ファイルを作成し、パーティションレイアウトを含む **MachineConfig** オブジェクトを含めます。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: primary
  name: 10_primary_storage_config
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      disks:
        - device: </dev/xxyN>
          partitions:
            - label: recovery
              startMiB: 32768
              sizeMiB: 16384
      filesystems:
        - device: /dev/disk/by-partlabel/recovery
          label: recovery
          format: xfs
```

2. **Custom-partitions.yaml** ファイルを保存して、**clusterconfigs/openshift** ディレクトリーにコピーします。

```
$ cp ~/<MachineConfig_manifest> ~/clusterconfigs/openshift
```

関連情報

- ベアメタルの設定
- パーティション命名スキーム

16.3.13. 非接続レジストリーの作成

インストールレジストリーのローカルコピーを使用して OpenShift Container Platform クラスターをインストールする必要がある場合があります。これにより、クラスターノードがインターネットにアクセスできないネットワーク上にあるため、ネットワークの効率が高まる可能性があります。

レジストリーのローカルまたはミラーリングされたコピーには、以下が必要です。

- レジストリーの証明書。これには、自己署名証明書を使用できます。
- システム上のコンテナーが提供する Web サーバー。
- 証明書およびローカルリポジトリーの情報が含まれる更新されたプルシークレット。

注記

レジストリーノードでの非接続レジストリーの作成はオプションです。レジストリーノードで切断されたレジストリーを作成する必要がある場合は、次のサブセクションをすべて完了する必要があります。

前提条件

- 非接続インストールのイメージのミラーリング 用にミラーレジストリーをすでに準備している場合は、非接続レジストリーを使用するように install-config.yaml ファイルを変更する へそのままスキップできます。

16.3.13.1. ミラーリングされたレジストリーをホストするためのレジストリーノードの準備

ベアメタルでミラー化されたレジストリーをホストする前に、次の手順を完了する必要があります。

手順

1. レジストリーノードでファイアウォールポートを開きます。

```
$ sudo firewall-cmd --add-port=5000/tcp --zone=libvirt  --permanent
```

```
$ sudo firewall-cmd --add-port=5000/tcp --zone=public   --permanent
```

```
$ sudo firewall-cmd --reload
```

2. レジストリーノードに必要なパッケージをインストールします。

```
$ sudo yum -y install python3 podman httpd httpd-tools jq
```

3. リポジトリー情報が保持されるディレクトリー構造を作成します。

```
$ sudo mkdir -p /opt/registry/{auth,certs,data}
```

16.3.13.2. 切断されたレジストリーの OpenShift Container Platform イメージリポジトリーのミラーリング

以下の手順を実行して、切断されたレジストリーの OpenShift Container Platform イメージリポジトリーをミラーリングします。

前提条件

- ミラーホストがインターネットにアクセスできる。
- ネットワークが制限された環境で使用するミラーレジストリーを設定し、設定した証明書および認証情報にアクセスできる。
- Red Hat OpenShift Cluster Manager からプルシークレット をダウンロードし、ミラーリポジトリーへの認証を組み込むように変更している。

手順

1. OpenShift Container Platform ダウンロード ページを確認し、インストールする必要のある OpenShift Container Platform のバージョンを判別し、Repository Tags ページで対応するタグを判別します。
2. 必要な環境変数を設定します。
    a. リリースバージョンをエクスポートします。

```
$ OCP_RELEASE=<release_version>
```

<release_version> について、インストールする OpenShift Container Platform のバージョンに対応するタグを指定します (例: **4.5.4**)。

b. ローカルレジストリー名とホストポートをエクスポートします。

```
$ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
```

<local_registry_host_name> については、ミラーレジストリーのレジストリードメイン名を指定し、**<local_registry_host_port>** については、コンテンツの送信に使用するポートを指定します。

c. ローカルリポジトリー名をエクスポートします。

```
$ LOCAL_REPOSITORY='<local_repository_name>'
```

<local_repository_name> については、**ocp4/openshift4** などのレジストリーに作成するリポジトリーの名前を指定します。

d. ミラーリングするリポジトリーの名前をエクスポートします。

```
$ PRODUCT_REPO='openshift-release-dev'
```

実稼働環境のリリースの場合には、**openshift-release-dev** を指定する必要があります。

e. パスをレジストリープルシークレットにエクスポートします。

```
$ LOCAL_SECRET_JSON='<path_to_pull_secret>'
```

<path_to_pull_secret> については、作成したミラーレジストリーのプルシークレットの絶対パスおよびファイル名を指定します。

f. リリースミラーをエクスポートします。

```
$ RELEASE_NAME="ocp-release"
```

実稼働環境のリリースについては、**ocp-release** を指定する必要があります。

g. クラスターのアーキテクチャーのタイプをエクスポートします。

```
$ ARCHITECTURE=<cluster_architecture> 1
```

1 **x86_64**、**aarch64**、**s390x**、または **ppc64le** など、クラスターのアーキテクチャーを指定します。

h. ミラーリングされたイメージをホストするためにディレクトリーへのパスをエクスポートします。

```
$ REMOVABLE_MEDIA_PATH=<path> 1
```

1 最初のスラッシュ (/) 文字を含む完全パスを指定します。

3. バージョンイメージをミラーレジストリーにミラーリングします。

- ミラーホストがインターネットにアクセスできない場合は、以下の操作を実行します。

  i. リムーバブルメディアをインターネットに接続しているシステムに接続します。

  ii. ミラーリングするイメージおよび設定マニフェストを確認します。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE} --dry-run
```

  iii. 直前のコマンドの出力の **imageContentSources** セクション全体を記録します。ミラーの情報はミラーリングされたリポジトリーに一意であり、インストール時に **imageContentSources** セクションを **install-config.yaml** ファイルに追加する必要があります。

  iv. イメージをリムーバブルメディア上のディレクトリーにミラーリングします。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}
```

  v. メディアをネットワークが制限された環境に移し、イメージをローカルコンテナーレジストリーにアップロードします。

```
$ oc image mirror -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} (1)
```

  (1) **REMOVABLE_MEDIA_PATH** の場合、イメージのミラーリング時に指定した同じパスを使用する必要があります。

- ローカルコンテナーレジストリーがミラーホストに接続されている場合は、以下の操作を実行します。

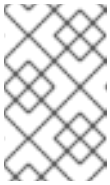
  i. 以下のコマンドを使用して、リリースイメージをローカルレジストリーに直接プッシュします。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
```

このコマンドは、リリース情報をダイジェストとしてプルします。その出力には、クラ

スターのインストール時に必要な **imageContentSources** データが含まれます。

ii. 直前のコマンドの出力の **imageContentSources** セクション全体を記録します。ミラーの情報はミラーリングされたリポジトリーに一意であり、インストール時に **imageContentSources** セクションを **install-config.yaml** ファイルに追加する必要があります。



注記

ミラーリングプロセス中にイメージ名に Quay.io のパッチが適用され、podman イメージにはブートストラップ仮想マシンのレジストリーに Quay.io が表示されます。

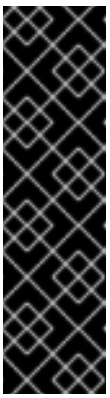
4. ミラーリングしたコンテンツに基づくインストールプログラムを作成するために、インストールプログラムを展開してリリースに固定します。

- ミラーホストがインターネットにアクセスできない場合は、以下のコマンドを実行します。

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-
baremetal-install "${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}"
```

- ローカルコンテナーレジストリーがミラーホストに接続されている場合は、以下のコマンドを実行します。

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-
baremetal-install "${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}"
```



重要

選択した OpenShift Container Platform のバージョンに適したイメージを確実に使用するために、ミラーリングしたコンテンツからインストールプログラムを展開する必要があります。

インターネット接続のあるマシンで、このステップを実行する必要があります。

非接続環境を使用している場合には、must-gather の一部として **--image** フラグを使用し、ペイロードイメージを参照します。

5. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの場合は、以下のコマンドを実行します。

```
$ openshift-baremetal-install
```

16.3.13.3. 非接続レジストリーを使用するように install-config.yaml ファイルを変更する

プロビジョナーノードでは、**install-config.yaml** ファイルは **pull-secret-update.txt** ファイルから新たに作成された pull-secret を使用する必要があります。**install-config.yaml** ファイルには、非接続レジストリーノードの証明書およびレジストリー情報も含まれる必要があります。

手順

1. 非接続レジストリーノードの証明書を **install-config.yaml** ファイルに追加します。

```
$ echo "additionalTrustBundle: |" >> install-config.yaml
```

証明書は **"additionalTrustBundle: |"** 行に従い、通常は 2 つのスペースで適切にインデントされる必要があります。

```
$ sed -e 's/^/  /' /opt/registry/certs/domain.crt >> install-config.yaml
```

2. レジストリーのミラー情報を **install-config.yaml** ファイルに追加します。

```
$ echo "imageContentSources:" >> install-config.yaml
```

```
$ echo "- mirrors:" >> install-config.yaml
```

```
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
```

registry.example.com をレジストリーの完全修飾ドメイン名に置き換えます。

```
$ echo "  source: quay.io/openshift-release-dev/ocp-release" >> install-config.yaml
```

```
$ echo "- mirrors:" >> install-config.yaml
```

```
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
```

registry.example.com をレジストリーの完全修飾ドメイン名に置き換えます。

```
$ echo "  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev" >> install-config.yaml
```

16.3.14. インストールの検証チェックリスト

❑ OpenShift Container Platform インストーラーが取得されている。

❑ OpenShift Container Platform インストーラーがデプロイメントされている。

❑ **install-config.yaml** の必須パラメーターが設定されている。

❑ **install-config.yaml** の **hosts** パラメーターが設定されている。

❑ **install-config.yaml** の **bmc** パラメーターが設定されている。

❑ **bmc address** フィールドで設定されている値の変換が適用されている。

❑ OpenShift Container Platform マニフェストが作成されている。

❑ (オプション) コンピュートノードにルーターがデプロイされている。

❑ (オプション) 切断されたレジストリーを作成している。

❑ (オプション) 非接続レジストリー設定が使用されている場合にこれを検証する。

16.3.15. OpenShift Container Platform インストーラーを使用したクラスターのデプロイ

OpenShift Container Platform インストーラーを実行します。

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create cluster
```

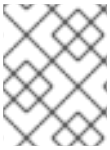
16.3.16. インストール後

デプロイメントプロセスで、**tail** コマンドを install ディレクトリーフォルダーの **.openshift_install.log** ログファイルに対して実行して、インストールの全体のステータスを確認できます。

```
$ tail -f /path/to/install-dir/.openshift_install.log
```

16.3.17. 静的 IP アドレス設定の検証

クラスターノードの DHCP 予約で無限リースが指定されている場合、インストーラーがノードを正常にプロビジョニングした後に、dispatcher スクリプトはノードのネットワーク設定をチェックします。ネットワーク設定に無限 DHCP リースが含まれているとスクリプトが判断すると、DHCP リースの IP アドレスを静的 IP アドレスとして使用して新規接続を作成します。



注記

dispatcher スクリプトは、クラスター内の他のノードのプロビジョニングの進行中に、正常にプロビジョニングされたノードで実行される場合があります。

ネットワーク設定が正しく機能していることを確認します。

手順

1. ノードのネットワークインターフェイス設定を確認してください。
2. DHCP サーバーをオフにし、OpenShift Container Platform ノードを再起動して、ネットワーク設定が適切に機能していることを確認します。

16.3.18. ベアメタルにクラスターを再インストールする準備

ベアメタルにクラスターを再インストールする前に、クリーンアップ操作を実行する必要があります。

手順

1. ブートストラップ、コントロールプレーンノード、およびコンピュートノードのディスクを削除するか、再フォーマットします。ハイパーバイザー環境で作業している場合は、削除したディスクを追加する必要があります。
2. 以前のインストールで生成されたアーティファクトを削除します。

```
$ cd ; /bin/rm -rf auth/ bootstrap.ign master.ign worker.ign metadata.json \
.openshift_install.log .openshift_install_state.json
```

3. 新しいマニフェストと Ignition 設定ファイルを生成します。詳細は、Kubernetes マニフェストおよび Ignition 設定ファイルの作成を参照してください。

4. インストールプログラムが作成した新規ブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これにより、以前の Ignition ファイルが上書きされます。

16.3.19. 関連情報

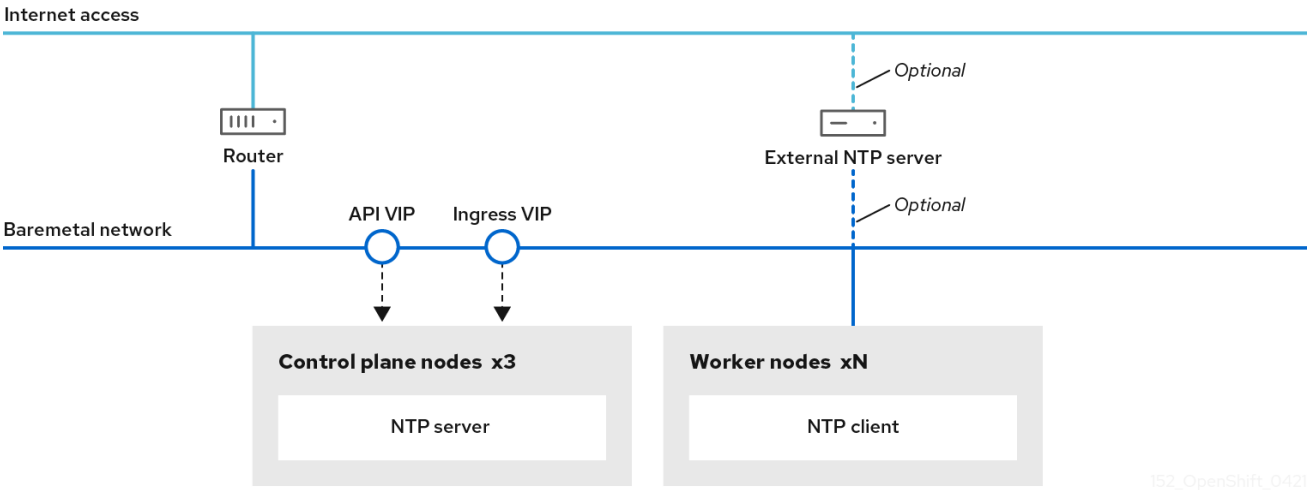
- OpenShift Container PlatformKubernetes マニフェストおよび Ignition 設定ファイルの作成
- 更新チャネルとリリースについて

16.4. INSTALLER-PROVISIONED のインストール後の設定

インストーラーでプロビジョニングされるクラスターを正常にデプロイしたら、以下のインストール後の手順を考慮してください。

16.4.1. オプション: 非接続クラスターの NTP 設定

OpenShift Container Platform は、クラスターノードに **chrony** Network Time Protocol (NTP) サービスをインストールします。次の手順を使用して、コントロールプレーンノードに NTP サーバーを設定し、デプロイが成功した後にコンピュートノードをコントロールプレーンノードの NTP クライアントとして設定します。



OpenShift Container Platform のノードは、適切に実行するために日付と時刻が一致している必要があります。コンピュートノードがコントロールプレーンノード上の NTP サーバーから日付と時刻を取得すると、ルーティング可能なネットワークに接続していないために上位層の NTP サーバーにアクセスできないクラスターのインストールと操作が可能になります。

手順

1. コントロールプレーンノードの **chrony.conf** ファイルのコンテンツを含む Butane 設定 (**99-master-chrony-conf-override.bu**) を作成します。

注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

Butane 設定例

-

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-master-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # Use public servers from the pool.ntp.org project.
          # Please consider joining the pool (https://www.pool.ntp.org/join.html).

          # The Machine Config Operator manages this file
          server openshift-master-0.<cluster-name>.<domain> iburst 1
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
          keyfile /etc/chrony.keys
          commandkey 1
          generatecommandkey
          noclientlog
          logchange 0.5
          logdir /var/log/chrony

          # Configure the control plane nodes to serve as local NTP servers
          # for all compute nodes, even if they are not in sync with an
          # upstream NTP server.

          # Allow NTP client access from the local network.
          allow all
          # Serve time even if not synchronized to a time source.
          local stratum 3 orphan
```

1 **<cluster-name>** はクラスターの名前に置き換え、**<domain>** は完全修飾ドメイン名に置き換える必要があります。

2. Butane を使用して、コントロールプレーンノードに配信される設定が含まれる **MachineConfig** オブジェクトファイル (**99-master-chrony-conf-override.yaml**) を生成します。

```
$ butane 99-master-chrony-conf-override.bu -o 99-master-chrony-conf-override.yaml
```

3. コントロールプレーンノード上の NTP サーバーを参照するコンピュートノードの **chrony.conf** ファイルの内容を含む Butane 設定 **99-worker-chrony-conf-override.bu** を作成します。

Butane 設定例

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # The Machine Config Operator manages this file.
          server openshift-master-0.<cluster-name>.<domain> iburst (1)
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
          keyfile /etc/chrony.keys
          commandkey 1
          generatecommandkey
          noclientlog
          logchange 0.5
          logdir /var/log/chrony
```

(1) **<cluster-name>** はクラスターの名前に置き換え、**<domain>** は完全修飾ドメイン名に置き換える必要があります。

4. Butane を使用して、ワーカーノードに配信される設定が含まれる **MachineConfig** オブジェクトファイル (**99-worker-chrony-conf-override.yaml**) を生成します。

```
$ butane 99-worker-chrony-conf-override.bu -o 99-worker-chrony-conf-override.yaml
```

5. **99-master-chrony-conf-override.yaml** ポリシーをコントロールプレーンノードに適用します。

```
$ oc apply -f 99-master-chrony-conf-override.yaml
```

出力例

```
machineconfig.machineconfiguration.openshift.io/99-master-chrony-conf-override created
```

6. **99-worker-chrony-conf-override.yaml** ポリシーをコンピュートノードに適用します。

```
$ oc apply -f 99-worker-chrony-conf-override.yaml
```

出力例

```
machineconfig.machineconfiguration.openshift.io/99-worker-chrony-conf-override created
```

7. 適用された NTP 設定のステータスを確認します。

```
$ oc describe machineconfigpool
```

16.4.2. インストール後のプロビジョニングネットワークの有効化

ベアメタルクラスター用のアシステッドインストーラーおよびインストーラーでプロビジョニングされるインストールは、**provisioning** ネットワークなしでクラスターをデプロイする機能を提供します。この機能は、概念実証クラスターや、各ノードのベースボード管理コントローラーが **baremetal** ネットワークを介してルーティング可能な場合に Redfish 仮想メディアのみを使用してデプロイするなどのシナリオ向けです。

Cluster Baremetal Operator (CBO) を使用してインストール後に **provisioning** ネットワークを有効にすることができます。

前提条件

- すべてのワーカーノードおよびコントロールプレーンノードに接続されている専用の物理ネットワークが存在する必要があります。
- ネイティブのタグなしの物理ネットワークを分離する必要があります。
- **provisioningNetwork** 設定が **Managed** に設定されている場合、ネットワークには DHCP サーバーを含めることはできません。
- OpenShift Container Platform 4.10 の **provisioningInterface** 設定を省略して、**bootMACAddress** 設定を使用できます。

手順

1. **provisioningInterface** 設定を設定する場合、まずクラスターノードのプロビジョニングインターフェイス名を特定します。たとえば、**eth0** または **eno1** などです。
2. クラスターノードの **provisioning** ネットワークインターフェイスで Preboot eXecution Environment (PXE) を有効にします。
3. **provisioning** ネットワークの現在の状態を取得して、これをプロビジョニングカスタムリソース (CR) ファイルに保存します。

```
$ oc get provisioning -o yaml > enable-provisioning-nw.yaml
```

4. プロビジョニング CR ファイルを変更します。

```
$ vim ~/enable-provisioning-nw.yaml
```

provisioningNetwork 設定までスクロールダウンして、これを **Disabled** から **Managed** に変更します。次に、**provisioningNetwork** 設定の後に、**provisioningIP**、**provisioningNetworkCIDR**、**provisioningDHCPRange**、**provisioningInterface**、および **watchAllNameSpaces** 設定を追加します。各設定に適切な値を指定します。

```
apiVersion: v1
items:
- apiVersion: metal3.io/v1alpha1
  kind: Provisioning
  metadata:
    name: provisioning-configuration
  spec:
    provisioningNetwork: 1
    provisioningIP: 2
    provisioningNetworkCIDR: 3
    provisioningDHCPRange: 4
    provisioningInterface: 5
    watchAllNameSpaces: 6
```

1. **provisioningNetwork** は、**Managed**、**Unmanaged**、または **Disabled** のいずれかになります。**Managed** に設定すると、Metal3 はプロビジョニングネットワークを管理し、CBO は設定済みの DHCP サーバーで Metal3 Pod をデプロイします。**Unmanaged** に設定すると、システム管理者は DHCP サーバーを手動で設定します。

2. **provisioningIP** は、DHCP サーバーおよび ironic がネットワークのプロビジョニングために使用する静的 IP アドレスです。この静的 IP アドレスは、DHCP 範囲外の **provisioning** 内でなければなりません。この設定を設定する場合は、**provisioning** ネットワークが **Disabled** の場合でも、有効な IP アドレスが必要です。静的 IP アドレスは metal3 Pod にバインドされます。metal3 Pod が失敗し、別のサーバーに移動する場合、静的 IP アドレスも新しいサーバーに移動します。

3. Classless Inter-Domain Routing (CIDR) アドレス。この設定を設定する場合は、**provisioning** ネットワークが **Disabled** の場合でも、有効な CIDR アドレスが必要です。例: **192.168.0.1/24**

4. DHCP の範囲。この設定は、 **Managed** プロビジョニングネットワークにのみ適用されます。**provisioning** ネットワークが **Disabled** の場合は、この設定を省略します。例: **192.168.0.64, 192.168.0.253**

5. クラスターノードの **provisioning** インターフェイス用の NIC 名。**provisioningInterface** 設定は、**Managed** および **Unmanaged** プロビジョニングネットワークにのみ適用されます。**provisioning** ネットワークが **Disabled** の場合に、**provisioningInterface** 設定が省略されます。代わりに **bootMACAddress** 設定を使用するように **provisioningInterface** 設定を省略します。

6. metal3 がデフォルトの **openshift-machine-api** namespace 以外の namespace を監視するようにするには、この設定を **true** に設定します。デフォルト値は **false** です。

5. 変更をプロビジョニング CR ファイルに保存します。

6. プロビジョニング CR ファイルをクラスターに適用します。

```
$ oc apply -f enable-provisioning-nw.yaml
```

16.5. クラスターの拡張

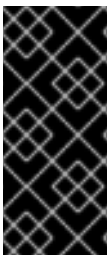
インストーラーでプロビジョニングされる OpenShift Container Platform クラスターのデプロイ後に、以下の手順を使用してワーカーノードの数を拡張することができます。それぞれの候補となるワーカーノードが前提条件を満たしていることを確認します。

注記

RedFish Virtual Media を使用してクラスターを拡張するには、最低限のファームウェア要件を満たす必要があります。RedFish Virtual Media を使用したクラスターの拡張時についての詳細は、**前提条件**セクションの**仮想メディアを使用したインストールのファームウェア要件**を参照してください。

16.5.1. ベアメタルノードの準備

クラスターを拡張するには、ノードに関連する IP アドレスを提供する必要があります。これは、静的設定または DHCP (動的ホスト設定プロトコル) サーバーを使用して行うことができます。DHCP サーバーを使用してクラスターを拡張する場合、各ノードには DHCP 予約が必要です。



IP アドレスの予約し、それらを静的 IP アドレスにする

一部の管理者は、各ノードの IP アドレスが DHCP サーバーがない状態で一定になるように静的 IP アドレスの使用を選択します。NMState で静的 IP アドレスを設定するには、追加の詳細について、OpenShift インストールの環境のセットアップセクションの (オプション) **install-config.yaml** でのホストネットワークインターフェイスの設定を参照してください。

ベアメタルノードを準備するには、プロビジョナーノードから以下の手順を実行する必要があります。

手順

1. **oc** バイナリーを取得します。

```
$ curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/openshift-client-linux-$VERSION.tar.gz | tar zxvf - oc
```

```
$ sudo cp oc /usr/local/bin
```

2. ベースボード管理コントローラー (BMC) を使用してベアメタルノードの電源を切り、オフになっていることを確認します。

3. ベアメタルノードのベースボード管理コントローラーのユーザー名およびパスワードを取得します。次に、ユーザー名とパスワードから **base64** 文字列を作成します。

```
$ echo -ne "root" | base64
```

```
$ echo -ne "password" | base64
```

4. ベアメタルノードの設定ファイルを作成します。静的設定または DHCP サーバーのどちらを使用しているかに応じて、次の例の **bmh.yaml** ファイルのいずれかを使用し、環境に合わせて YAML の値を置き換えます。

```
$ vim bmh.yaml
```

- **静的設定 bmh.yaml** :

```
---
apiVersion: v1 (1)
kind: Secret
metadata:
 name: openshift-worker-<num>-network-config-secret (2)
 namespace: openshift-machine-api
type: Opaque
stringData:
 nmstate: | (3)
  interfaces: (4)
  - name: <nic1_name> (5)
    type: ethernet
    state: up
    ipv4:
      address:
      - ip: <ip_address> (6)
        prefix-length: 24
      enabled: true
  dns-resolver:
    config:
      server:
      - <dns_ip_address> (7)
  routes:
    config:
    - destination: 0.0.0.0/0
      next-hop-address: <next_hop_ip_address> (8)
      next-hop-interface: <next_hop_nic1_name> (9)
---
apiVersion: v1
kind: Secret
metadata:
  name: openshift-worker-<num>-bmc-secret (10)
  namespace: openshift-machine-api
type: Opaque
data:
  username: <base64_of_uid> (11)
  password: <base64_of_pwd> (12)
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: openshift-worker-<num> (13)
  namespace: openshift-machine-api
spec:
  online: True
  bootMACAddress: <nic1_mac_address> (14)
  bmc:
    address: <protocol>://<bmc_url> (15)
    credentialsName: openshift-worker-<num>-bmc-secret (16)
    disableCertificateVerification: True (17)
    username: <bmc_username> (18)
```

```
    password: <bmc_password> 19
  rootDeviceHints:
    deviceName: <root_device_hint> 20
  preprovisioningNetworkDataName: openshift-worker-<num>-network-config-secret 21
```

1 新しく作成されたノードのネットワークインターフェイスを設定するには、ネットワーク設定を含むシークレットの名前を指定します。**nmstate** 構文に従って、ノードのネットワーク設定を定義します。NMState 構文の設定の詳細については、(オプション) install-config.yaml ファイルでのホストネットワークインターフェイスの設定を参照してください。

2 10 13 16 **name** フィールド、**credentialsName** フィールド、および **preprovisioningNetworkDataName** フィールドで、ベアメタルノードのワーカー数の **<num>** を置き換えます。

3 NMState YAML 構文を追加して、ホストインターフェイスを設定します。

4 オプション: **nmstate** を使用してネットワークインターフェイスを設定しており、インターフェイスを無効にする場合は、次のように IP アドレスを **enabled: false** に設定して **state: up** を設定します。

```
---
  interfaces:
  - name: <nic_name>
    type: ethernet
    state: up
    ipv4:
      enabled: false
    ipv6:
      enabled: false
```

5 6 7 8 9 **<nic1_name>**、**<ip_address>**、**<dns_ip_address>**、**<next_hop_ip_address>**、および **<next_hop_nic1_name>** を適切な値に置き換えます。

11 12 **<base64_of_uid>** と **<base64_of_pwd>** を、ユーザー名とパスワードの base64 文字列に置き換えます。

14 **<nic1_mac_address>** を、ベアメタルノードの最初の NIC の MAC アドレスに置き換えます。追加の BMC 設定オプションについては、BMC アドレス指定のセクションを参照してください。

15 **<protocol>** を、IPMI、 RedFish その他の BMC プロトコルに置き換えマス。**<bmc_url>** を、ベアメタルノードのベースボード管理コントローラーの URL に置き換えます。

17 証明書の検証をスキップするには、**disableCertificateVerification** を true に設定します。

18 19 **<bmc_username>** と **<bmc_password>** を BMC ユーザー名とパスワードの文字列に置き換えます。

20 オプション: root デバイスヒントを指定する場合は、**<root_device_hint>** をデバイスパスに置き換えます。

21 オプション: 新しく作成されたノードのネットワークインターフェイスを設定した場合

は、BareMetalHost CR の **preprovisioningNetworkDataName** にネットワーク設定のシークレット名を指定します。

- **DHCP 設定 bmh.yaml** :

```
---
apiVersion: v1
kind: Secret
metadata:
  name: openshift-worker-<num>-bmc-secret (1)
  namespace: openshift-machine-api
type: Opaque
data:
  username: <base64_of_uid> (2)
  password: <base64_of_pwd> (3)
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: openshift-worker-<num> (4)
  namespace: openshift-machine-api
spec:
  online: True
  bootMACAddress: <nic1_mac_address> (5)
  bmc:
    address: <protocol>://<bmc_url> (6)
    credentialsName: openshift-worker-<num>-bmc-secret (7)
    disableCertificateVerification: True (8)
    username: <bmc_username> (9)
    password: <bmc_password> (10)
  rootDeviceHints:
    deviceName: <root_device_hint> (11)
  preprovisioningNetworkDataName: openshift-worker-<num>-network-config-secret (12)
```

(1) (4) (7) **name** フィールド、**credentialsName** フィールド、および **preprovisioningNetworkDataName** フィールドで、ベアメタルノードのワーカー数の **<num>** を置き換えます。

(2) (3) **<base64_of_uid>** と **<base64_of_pwd>** を、ユーザー名とパスワードの base64 文字列に置き換えます。

(5) **<nic1_mac_address>** を、ベアメタルノードの最初の NIC の MAC アドレスに置き換えます。追加の BMC 設定オプションについては、BMC アドレス指定のセクションを参照してください。

(6) **<protocol>** を、IPMI、 RedFish その他の BMC プロトコルに置き換えマス。**<bmc_url>** を、ベアメタルノードのベースボード管理コントローラーの URL に置き換えます。

(8) 証明書の検証をスキップするには、**disableCertificateVerification** を true に設定します。

(9) (10) **<bmc_username>** と **<bmc_password>** を BMC ユーザー名とパスワードの文字列に置き換えます。

11 オプション: root デバイスヒントを指定する場合は、**<root_device_hint>** をデバイスパスに置き換えます。

12 オプション: 新しく作成されたノードのネットワークインターフェイスを設定した場合は、BareMetalHost CR の **preprovisioningNetworkDataName** にネットワーク設定のシークレット名を指定します。

注記

既存のベアメタルノードの MAC アドレスが、プロビジョニングしようとしているベアメタルホストの MAC アドレスと一致する場合、Ironic インストールは失敗します。ホストの登録、検査、クリーニング、または他の Ironic 手順が失敗する場合、ベアメタル Operator はインストールを継続的に再試行します。詳細については、ホストの重複した MAC アドレスの診断を参照してください。

5. ベアメタルノードを作成します。

```
$ oc -n openshift-machine-api create -f bmh.yaml
```

出力例

```
secret/openshift-worker-<num>-network-config-secret created
secret/openshift-worker-<num>-bmc-secret created
baremetalhost.metal3.io/openshift-worker-<num> created
```

ここで、**<num>** はワーカー数です。

6. ベアメタルノードの電源をオンにし、これを検査します。

```
$ oc -n openshift-machine-api get bmh openshift-worker-<num>
```

ここで、**<num>** はワーカーノード数です。

出力例

```
NAME                 STATE      CONSUMER   ONLINE   ERROR
openshift-worker-<num>  available            true
```

注記

ワーカーノードがクラスターに参加できるようにするには、**machineset** オブジェクトを **BareMetalHost** オブジェクトの数にスケーリングします。ノードは手動または自動でスケーリングできます。ノードを自動的にスケーリングするには、**machineset** に **metal3.io/autoscale-to-hosts** アノテーションを使用します。

関連情報

- NMState 構文の設定に関する詳細は、(オプション) install-config.yaml ファイルでのホストネットワークインターフェイスの設定 を参照してください。

- マシンの自動スケーリングの詳細は、利用可能なベアメタルホストの数へのマシンの自動スケーリング を参照してください。

16.5.2. ベアメタルコントロールプレーンノードの交換

以下の手順を使用して、インストーラーによってプロビジョニングされた OpenShift Container Platform コントロールプレーンノードを置き換えます。

重要

既存のコントロールプレーンホストから **BareMetalHost** オブジェクト定義を再利用する場合は、**externallyProvisioned** フィールドを **true** に設定したままにしないでください。

既存のコントロールプレーン **BareMetalHost** オブジェクトが、OpenShift Container Platform インストールプログラムによってプロビジョニングされた場合には、**externallyProvisioned** フラグが **true** に設定されている可能性があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- etcd のバックアップを取得している。

重要

問題が発生した場合にクラスターを復元できるように、この手順を実行する前に etcd のバックアップを作成してください。etcd バックアップの作成に関する詳細は、**関連情報** セクションを参照してください。

手順

1. Bare Metal Operator が使用可能であることを確認します。

```
$ oc get clusteroperator baremetal
```

出力例

```
NAME       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
baremetal  4.16      True        False         False      3d15h
```

2. 古い **BareMetalHost** オブジェクトおよび **Machine** オブジェクトを削除します。

```
$ oc delete bmh -n openshift-machine-api <host_name>
$ oc delete machine -n openshift-machine-api <machine_name>
```

<host_name> をホストの名前に、**<machine_name>** をマシンの名前に置き換えます。マシン名は **CONSUMER** フィールドの下に表示されます。

BareMetalHost オブジェクトと **Machine** オブジェクトを削除すると、マシンコントローラーは **Node** オブジェクトを自動的に削除します。

3. 新しい **BareMetalHost** オブジェクトとシークレットを作成して BMC 認証情報を保存します。

```
$ cat <<EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: control-plane-<num>-bmc-secret (1)
  namespace: openshift-machine-api
data:
  username: <base64_of_uid> (2)
  password: <base64_of_pwd> (3)
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: control-plane-<num> (4)
  namespace: openshift-machine-api
spec:
  automatedCleaningMode: disabled
  bmc:
    address: <protocol>://<bmc_ip> (5)
    credentialsName: control-plane-<num>-bmc-secret (6)
  bootMACAddress: <NIC1_mac_address> (7)
  bootMode: UEFI
  externallyProvisioned: false
  online: true
EOF
```

(1) (4) (6) **name** フィールドと **credentialsName** フィールドにあるベアメタルノードのコントロールプレーン数の **<num>** を置き換えます。

(2) **<base64_of_uid>** を、ユーザー名の **base64** 文字列に置き換えます。

(3) **<base64_of_pwd>>** を、パスワードの **base64** 文字列に置き換えます。

(5) **<protocol>** を **redfish**、**redfish-virtualmedia**、**idrac-virtualmedia** などの BMC プロトコルに置き換えます。**<bmc_ip>** を、ベアメタルノードのベースボード管理コントローラー (BMC) の IP アドレスに置き換えます。その他の BMC 設定オプションについては、**関連情報** セクションのBMC アドレス指定を参照してください。

(7) **<NIC1_mac_address>** を、ベアメタルの最初の NIC の MAC アドレスに置き換えます。

検査が完了すると、**BareMetalHost** オブジェクトが作成され、プロビジョニングできるようになります。

4. 利用可能な **BareMetalHost** オブジェクトを表示します。

```
$ oc get bmh -n openshift-machine-api
```

出力例

```
NAME                         STATE                  CONSUMER          ONLINE   ERROR   AGE
control-plane-1.example.com  available              control-plane-1   true             1h10m
control-plane-2.example.com  externally provisioned  control-plane-2   true             4h53m
```

```
control-plane-3.example.com   externally provisioned   control-plane-3          true
4h53m
compute-1.example.com         provisioned             compute-1-ktmmx          true
4h53m
compute-1.example.com         provisioned             compute-2-l2zmb          true
4h53m
```

コントロールプレーンノード用の **MachineSet** オブジェクトがないため、代わりに **Machine** オブジェクトを作成する必要があります。別のコントロールプレーン **Machine** オブジェクトから **providerSpec** をコピーできます。

5. **Machine** オブジェクトを作成します。

```
$ cat <<EOF | oc apply -f -
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  annotations:
    metal3.io/BareMetalHost: openshift-machine-api/control-plane-<num> 1
  labels:
    machine.openshift.io/cluster-api-cluster: control-plane-<num> 2
    machine.openshift.io/cluster-api-machine-role: master
    machine.openshift.io/cluster-api-machine-type: master
  name: control-plane-<num> 3
  namespace: openshift-machine-api
spec:
  metadata: {}
  providerSpec:
    value:
      apiVersion: baremetal.cluster.k8s.io/v1alpha1
      customDeploy:
        method: install_coreos
      hostSelector: {}
      image:
        checksum: ""
        url: ""
      kind: BareMetalMachineProviderSpec
      metadata:
        creationTimestamp: null
      userData:
        name: master-user-data-managed
EOF
```

1 2 3 **name** フィールド、**labels** フィールド、および **annotations** フィールドにあるベアメタルノードのコントロールプレーン数の **<num>** を置き換えます。

6. **BareMetalHost** オブジェクトを表示するには、次のコマンドを実行します。

```
$ oc get bmh -A
```

出力例

```
NAME                        STATE         CONSUMER          ONLINE   ERROR   AGE
control-plane-1.example.com   provisioned   control-plane-1   true             2h53m
```

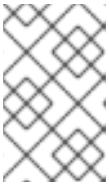
```
control-plane-2.example.com   externally provisioned   control-plane-2          true
5h53m
control-plane-3.example.com   externally provisioned   control-plane-3          true
5h53m
compute-1.example.com         provisioned              compute-1-ktmmx          true
5h53m
compute-2.example.com         provisioned              compute-2-l2zmb          true
5h53m
```

7. RHCOS のインストール後、**BareMetalHost** がクラスターに追加されていることを確認します。

```
$ oc get nodes
```

出力例

```
NAME                         STATUS      ROLES     AGE   VERSION
control-plane-1.example.com   available   master    4m2s  v1.29.4
control-plane-2.example.com   available   master    141m  v1.29.4
control-plane-3.example.com   available   master    141m  v1.29.4
compute-1.example.com         available   worker    87m   v1.29.4
compute-2.example.com         available   worker    87m   v1.29.4
```



注記

新しいコントロールプレーンノードの交換後、新しいノードで実行されている etcd Pod は **crashloopback** ステータスになります。詳細は、**関連情報** セクションの正常でない etcd メンバーの置き換えを参照してください。

関連情報

- 正常でない etcd メンバーの置き換え
- etcd のバックアップ
- ベアメタルの設定
- BMC アドレス指定

16.5.3. ベアメタルネットワークに仮想メディアを使用してデプロイメントする準備

provisioning ネットワークが有効で、**baremetal** ネットワークで Virtual Media を使用してクラスターを拡張する場合は、以下の手順を使用します。

前提条件

- **baremetal** ネットワークおよび **provisioning** ネットワークを使用する既存のクラスターがあります。

手順

1. **provisioning** カスタムリソース (CR) を編集して、 **baremetal** ネットワーク上の仮想メディアを使用したデプロイを有効にします。

```
oc edit provisioning
```

```
apiVersion: metal3.io/v1alpha1
kind: Provisioning
metadata:
  creationTimestamp: "2021-08-05T18:51:50Z"
  finalizers:
  - provisioning.metal3.io
  generation: 8
  name: provisioning-configuration
  resourceVersion: "551591"
  uid: f76e956f-24c6-4361-aa5b-feaf72c5b526
spec:
  provisioningDHCPRange: 172.22.0.10,172.22.0.254
  provisioningIP: 172.22.0.3
  provisioningInterface: enp1s0
  provisioningNetwork: Managed
  provisioningNetworkCIDR: 172.22.0.0/24
  virtualMediaViaExternalNetwork: true (1)
status:
  generations:
  - group: apps
    hash: ""
    lastGeneration: 7
    name: metal3
    namespace: openshift-machine-api
    resource: deployments
  - group: apps
    hash: ""
    lastGeneration: 1
    name: metal3-image-cache
    namespace: openshift-machine-api
    resource: daemonsets
  observedGeneration: 8
  readyReplicas: 0
```

(1) **virtualMediaViaExternalNetwork: true** を **provisioning** CR に追加します。

2. イメージ URL が存在する場合は、**machineset** を編集して API VIP アドレスを使用します。この手順は、バージョン 4.9 以前でインストールされたクラスターにのみ適用されます。

```
oc edit machineset
```

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: "2021-08-05T18:51:52Z"
  generation: 11
  labels:
    machine.openshift.io/cluster-api-cluster: ostest-hwmdt
    machine.openshift.io/cluster-api-machine-role: worker
    machine.openshift.io/cluster-api-machine-type: worker
  name: ostest-hwmdt-worker-0
  namespace: openshift-machine-api
```

```
    resourceVersion: "551513"
    uid: fad1c6e0-b9da-4d4a-8d73-286f78788931
  spec:
    replicas: 2
    selector:
      matchLabels:
        machine.openshift.io/cluster-api-cluster: ostest-hwmdt
        machine.openshift.io/cluster-api-machineset: ostest-hwmdt-worker-0
    template:
      metadata:
        labels:
          machine.openshift.io/cluster-api-cluster: ostest-hwmdt
          machine.openshift.io/cluster-api-machine-role: worker
          machine.openshift.io/cluster-api-machine-type: worker
          machine.openshift.io/cluster-api-machineset: ostest-hwmdt-worker-0
      spec:
        metadata: {}
        providerSpec:
          value:
            apiVersion: baremetal.cluster.k8s.io/v1alpha1
            hostSelector: {}
            image:
              checksum: http:/172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2.
<md5sum> (1)
              url: http://172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2 (2)
            kind: BareMetalMachineProviderSpec
            metadata:
              creationTimestamp: null
            userData:
              name: worker-user-data
  status:
    availableReplicas: 2
    fullyLabeledReplicas: 2
    observedGeneration: 11
    readyReplicas: 2
    replicas: 2
```

(1) API VIP アドレスを使用するように **checksum** URL を編集します。

(2) **url** URL を編集して API VIP アドレスを使用します。

16.5.4. クラスター内の新しいホストをプロビジョニングする際の重複する MAC アドレスの診断

クラスター内の既存のベアメタルノードの MAC アドレスが、クラスターに追加しようとしているベアメタルホストの MAC アドレスと一致する場合、ベアメタル Operator はホストを既存のノードに関連付けます。ホストの登録、検査、クリーニング、または他の Ironic 手順が失敗する場合、ベアメタル Operator はインストールを継続的に再試行します。障害が発生したベアメタルホストの登録エラーが表示されます。

openshift-machine-api namespace で実行されているベアメタルホストを調べることで、重複する MAC アドレスを診断できます。

前提条件

- ベアメタルに OpenShift Container Platform クラスターをインストールします。
- OpenShift Container Platform CLI (**oc**) をインストールします。
- **cluster-admin** 権限を持つユーザーとしてログインしている。

手順

プロビジョニングに失敗したベアメタルホストが既存のノードと同じ MAC アドレスを持つかどうかを判断するには、以下を実行します。

1. **openshift-machine-api** namespace で実行されているベアメタルホストを取得します。

```
$ oc get bmh -n openshift-machine-api
```

出力例

```
NAME                STATUS   PROVISIONING STATUS      CONSUMER
openshift-master-0   OK      externally provisioned   openshift-zpwpq-master-0
openshift-master-1   OK      externally provisioned   openshift-zpwpq-master-1
openshift-master-2   OK      externally provisioned   openshift-zpwpq-master-2
openshift-worker-0   OK      provisioned              openshift-zpwpq-worker-0-lv84n
openshift-worker-1   OK      provisioned              openshift-zpwpq-worker-0-zd8lm
openshift-worker-2   error   registering
```

2. 障害が発生したホストのステータスに関する詳細情報を表示するには、**<bare_metal_host_name>** をホストの名前に置き換えて、以下のコマンドを実行します。

```
$ oc get -n openshift-machine-api bmh <bare_metal_host_name> -o yaml
```

出力例

```
...
status:
  errorCount: 12
  errorMessage: MAC address b4:96:91:1d:7c:20 conflicts with existing node openshift-
worker-1
  errorType: registration error
...
```

16.5.5. ベアメタルノードのプロビジョニング

ベアメタルノードをプロビジョニングするには、プロビジョナーノードから以下の手順を実行する必要があります。

手順

1. ベアメタルノードをプロビジョニングする前に、**STATE** が **available** であることを確認してください。

```
$  oc -n openshift-machine-api get bmh openshift-worker-<num>
```

ここで、**<num>** はワーカーノード数です。

```
NAME              STATE     ONLINE ERROR  AGE
openshift-worker  available true         34h
```

2. ワーカーノードの数を取得します。

```
$ oc get nodes
```

```
NAME                                         STATUS   ROLES         AGE     VERSION
openshift-master-1.openshift.example.com     Ready    master        30h     v1.29.4
openshift-master-2.openshift.example.com     Ready    master        30h     v1.29.4
openshift-master-3.openshift.example.com     Ready    master        30h     v1.29.4
openshift-worker-0.openshift.example.com     Ready    worker        30h     v1.29.4
openshift-worker-1.openshift.example.com     Ready    worker        30h     v1.29.4
```

3. コンピュートマシンセットを取得します。

```
$ oc get machinesets -n openshift-machine-api
```

```
NAME                          DESIRED   CURRENT   READY   AVAILABLE   AGE
...
openshift-worker-0.example.com     1         1         1       1          55m
openshift-worker-1.example.com     1         1         1       1          55m
```

4. ワーカーノードの数を 1 つずつ増やします。

```
$ oc scale --replicas=<num> machineset <machineset> -n openshift-machine-api
```

<num> を、ワーカーノードの新たな数に置き換えます。**<machineset>** を、前の手順で設定したコンピューティングマシンの名前に置き換えます。

5. ベアメタルノードのステータスを確認します。

```
$ oc -n openshift-machine-api get bmh openshift-worker-<num>
```

ここで、**<num>** はワーカーノード数です。STATE が **ready** から **provisioning** に変わります。

```
NAME                  STATE          CONSUMER                      ONLINE   ERROR
openshift-worker-<num>  provisioning     openshift-worker-<num>-65tjz      true
```

provisioning ステータスは、OpenShift Container Platform クラスターがノードをプロビジョニングするまでそのままになります。この場合、30 分以上の時間がかかる場合があります。ノードがプロビジョニングされると、状態は **provisioned** に変わります。

```
NAME                  STATE          CONSUMER                      ONLINE   ERROR
openshift-worker-<num>  provisioned      openshift-worker-<num>-65tjz      true
```

6. プロビジョニングが完了したら、ベアメタルノードが準備状態であることを確認します。

```
$ oc get nodes
```

```
NAME                                         STATUS   ROLES   AGE     VERSION
openshift-master-1.openshift.example.com     Ready    master  30h     v1.29.4
```

```
openshift-master-2.openshift.example.com      Ready    master  30h     v1.29.4
openshift-master-3.openshift.example.com      Ready    master  30h     v1.29.4
openshift-worker-0.openshift.example.com      Ready    worker  30h     v1.29.4
openshift-worker-1.openshift.example.com      Ready    worker  30h     v1.29.4
openshift-worker-<num>.openshift.example.com  Ready    worker  3m27s   v1.29.4
```

kubelet を確認することもできます。

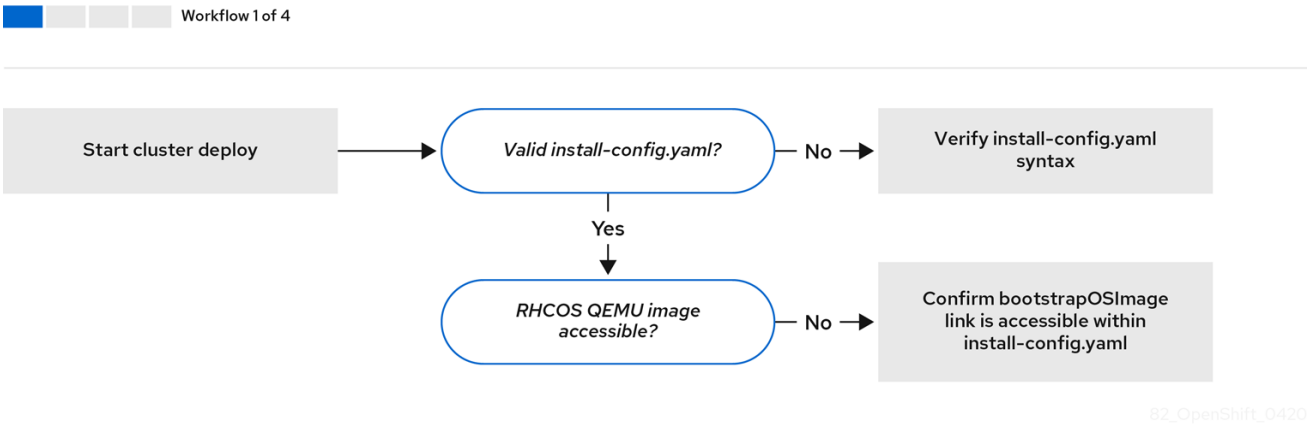
```
$ ssh openshift-worker-<num>
```

```
[kni@openshift-worker-<num>]$ journalctl -fu kubelet
```

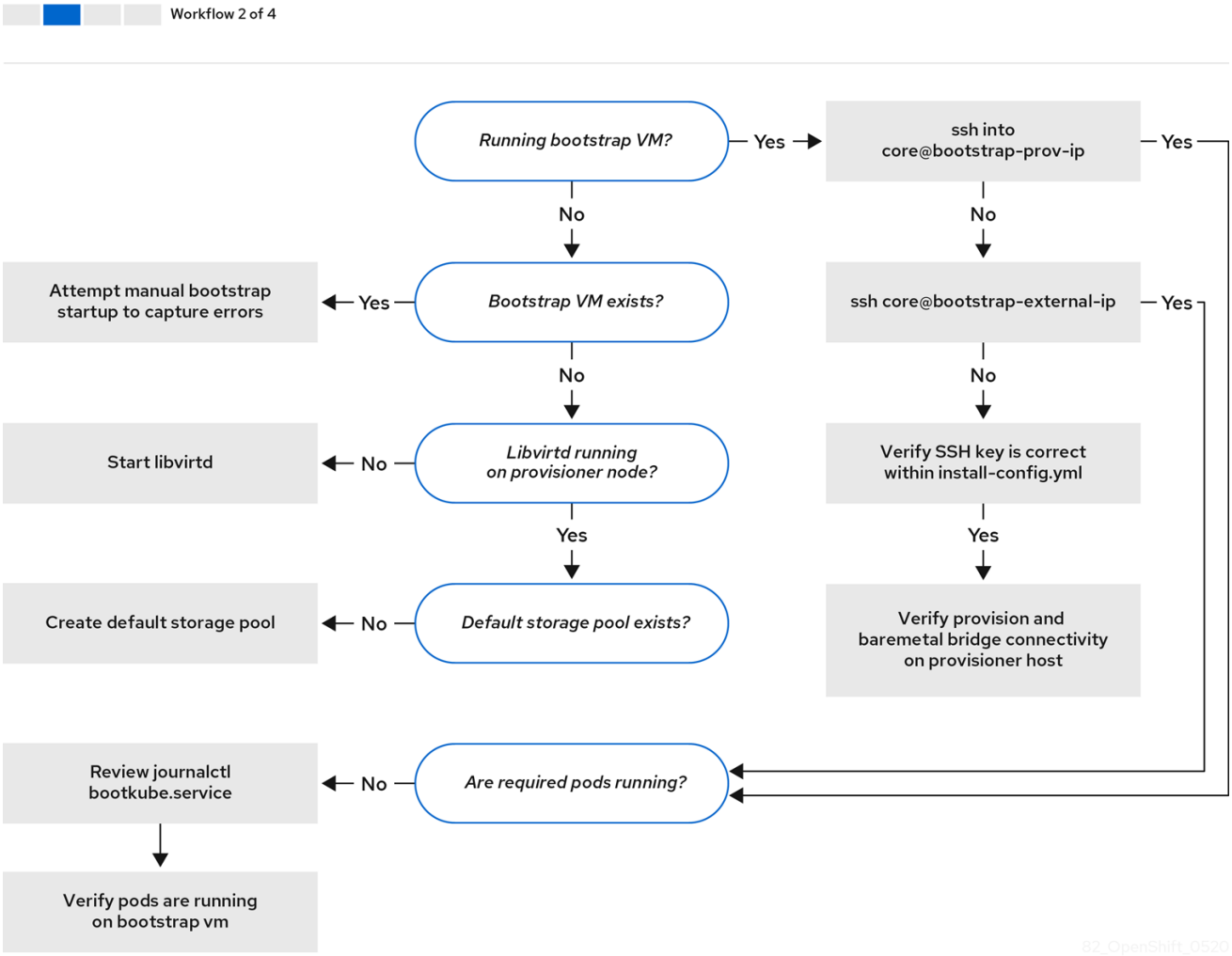
16.6. トラブルシューティング

16.6.1. インストールプログラムワークフローのトラブルシューティング

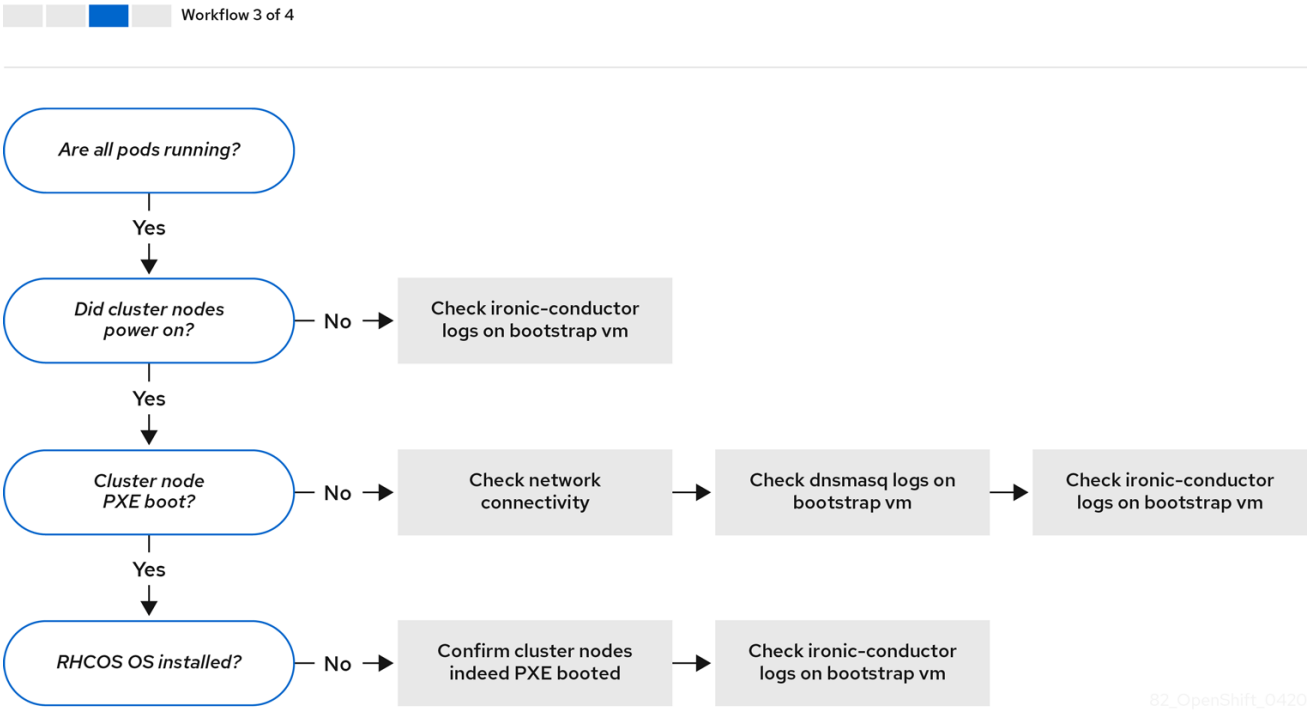
インストール環境のトラブルシューティングを行う前に、ベアメタルへのインストーラーでプロビジョニングされるインストールの全体的なフローを理解することは重要です。以下の図は、環境におけるステップごとのトラブルシューティングフローを示しています。



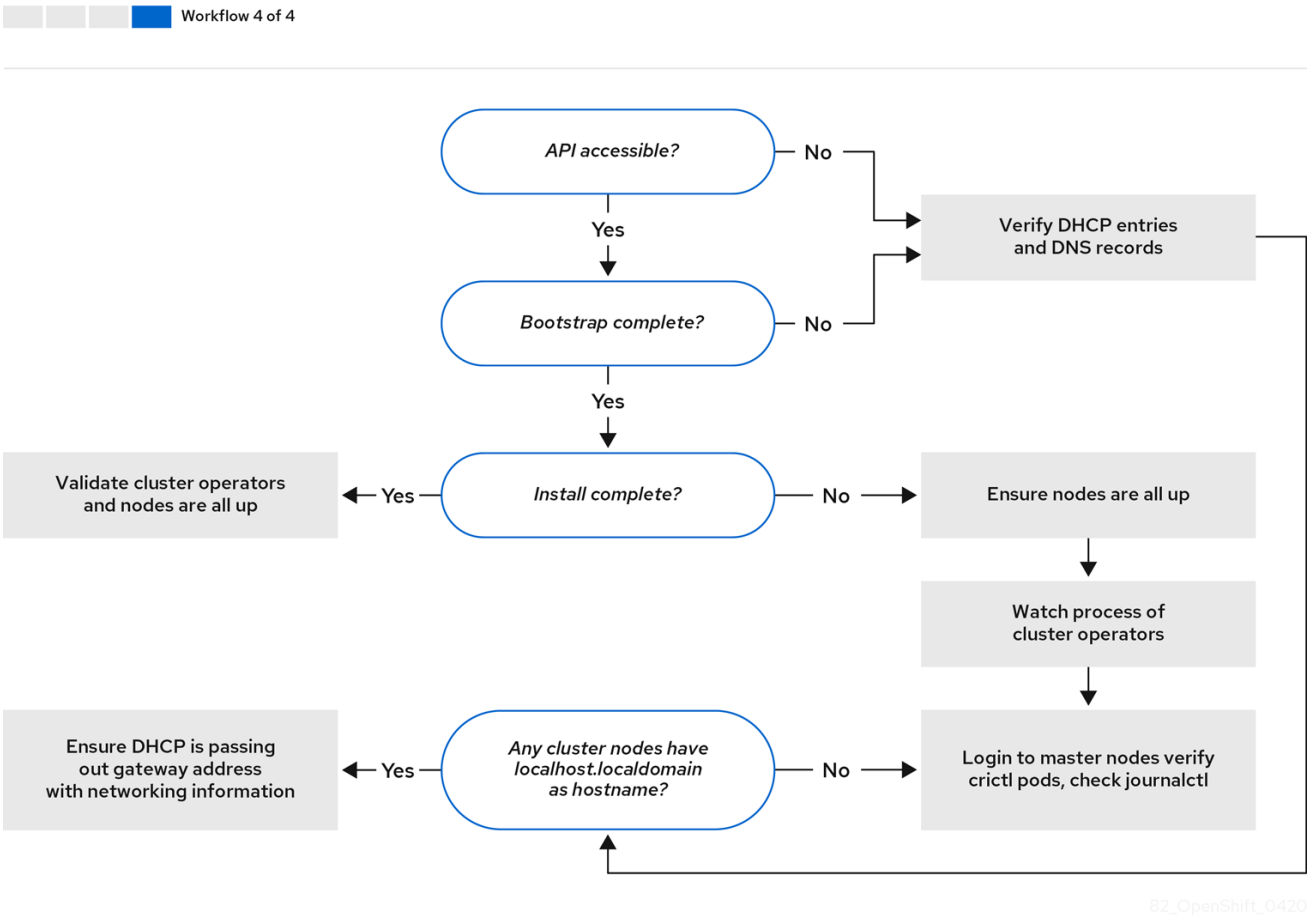
ワークフロー 1/4 は、**install-config.yaml** ファイルにエラーがある場合や Red Hat Enterprise Linux CoreOS (RHCOS) イメージにアクセスできない場合のトラブルシューティングのワークフローを説明しています。トラブルシューティングについての提案は、Troubleshooting **install-config.yaml** を参照してください。



ワークフロー 2/4 は、ブートストラップ仮想マシンの問題、クラスターノードを起動できないブートストラップ仮想マシン、および ログの検査 についてのトラブルシューティングのワークフローを説明しています。**provisioning** ネットワークなしに OpenShift Container Platform クラスターをインストールする場合は、このワークフローは適用されません。



ワークフロー 3/4 は、PXE ブートしないクラスターノード のトラブルシューティングのワークフローを説明しています。RedFish 仮想メディアを使用してインストールする場合、インストールプログラムによるノードのデプロイに必要な最小ファームウェア要件を各ノードが満たしている必要があります。詳細は、**前提条件**セクションの**仮想メディアを使用したインストールのファームウェア要件**を参照してください。



ワークフロー 4/4 は、アクセスできない API から 検証済みのインストール までのトラブルシューティングワークフローを示しています。

16.6.2. install-config.yaml のトラブルシューティング

install-config.yaml 設定ファイルは、OpenShift Container Platform クラスターの一部であるすべてのノードを表します。このファイルには、**apiVersion**、**baseDomain**、**imageContentSources**、および仮想 IP アドレスのみで設定されるがこれらに制限されない必要なオプションが含まれます。OpenShift Container Platform クラスターのデプロイメントの初期段階でエラーが発生した場合、エラーは **install-config.yaml** 設定ファイルにある可能性があります。

手順

1. YAML-tips のガイドラインを使用します。
2. syntax-check を使用して YAML 構文が正しいことを確認します。
3. Red Hat Enterprise Linux CoreOS (RHCOS) QEMU イメージが適切に定義され、**install-config.yaml** で提供される URL 経由でアクセスできることを確認します。以下に例を示します。

```
$ curl -s -o /dev/null -I -w "%{http_code}\n" http://webserver.example.com:8080/rhcos-44.81.202004250133-0-qemu.<architecture>.qcow2.gz?sha256=7d884b46ee54fe87bbc3893bf2aa99af3b2d31f2e19ab5529c60636fbd0f1ce7
```

出力が **200** の場合、ブートストラップ仮想マシンイメージを保存する Web サーバーからの有効な応答があります。

16.6.3. ブートストラップ仮想マシンの問題のトラブルシューティング

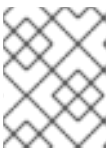
OpenShift Container Platform インストールプログラムは、OpenShift Container Platform クラスターノードのプロビジョニングを処理するブートストラップノードの仮想マシンを起動します。

手順

1. インストールプログラムをトリガー後の約 10 分から 15 分後に、**virsh** コマンドを使用してブートストラップ仮想マシンが機能していることを確認します。

```
$ sudo virsh list
```

```
 Id    Name                           State
----------------------------------------------------
 12    openshift-xf6fq-bootstrap      running
```



注記

ブートストラップ仮想マシンの名前は常にクラスター名で始まり、その後にランダムな文字セットが続き、bootstrap という単語で終わります。

2. 10 - 15 分経ってもブートストラップ仮想マシンが実行されない場合は、次のコマンドを実行して、システム上で **libvirtd** が実行されていることを確認します。

```
$ systemctl status libvirtd
```

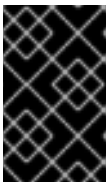
```
–
● libvirtd.service - Virtualization daemon
  Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2020-03-03 21:21:07 UTC; 3 weeks 5 days ago
    Docs: man:libvirtd(8)
          https://libvirt.org
Main PID: 9850 (libvirtd)
   Tasks: 20 (limit: 32768)
  Memory: 74.8M
  CGroup: /system.slice/libvirtd.service
          ├─ 9850 /usr/sbin/libvirtd
```

ブートストラップ仮想マシンが動作している場合は、これにログインします。

3. **virsh console** コマンドを使用して、ブートストラップ仮想マシンの IP アドレスを見つけます。

```
$ sudo virsh console example.com
```

```
Connected to domain example.com
Escape character is ^]
Red Hat Enterprise Linux CoreOS 43.81.202001142154.0 (Ootpa) 4.3
SSH host key: SHA256:BRWJktXZgQQRY5zjuAV0IKZ4WM7i4TiUyMVanqu9Pqg (ED25519)
SSH host key: SHA256:7+iKGA7VtG5szmk2jB5gl/5EZ+SNcJ3a2g23o0lnIio (ECDSA)
SSH host key: SHA256:DH5VWhvhvagOTaLsYiVNse9ca+ZSW/30OOMed8rIGOc (RSA)
ens3:  fd35:919d:4042:2:c7ed:9a9f:a9ec:7
ens4: 172.22.0.2 fe80::1d05:e52e:be5d:263f
localhost login:
```



重要

provisioning ネットワークなしで OpenShift Container Platform クラスターをデプロイする場合、**172.22.0.2** などのプライベート IP アドレスではなく、パブリック IP アドレスを使用する必要があります。

4. IP アドレスを取得したら、**ssh** コマンドを使用してブートストラップ仮想マシンにログインします。

注記

直前の手順のコンソール出力では、**ens3** で提供される IPv6 IP アドレスまたは **ens4** で提供される IPv4 IP を使用できます。

```
$ ssh core@172.22.0.2
```

ブートストラップ仮想マシンへのログインに成功しない場合は、以下いずれかのシナリオが発生した可能性があります。

- **172.22.0.0/24** ネットワークにアクセスできない。プロビジョナーと **provisioning** ネットワークブリッジ間のネットワーク接続を確認します。この問題は、**provisioning** ネットワークを使用している場合に発生することがあります。

- パブリックネットワーク経由でブートストラップ仮想マシンにアクセスできない。**baremetal** ネットワークで SSH を試行する際に、**provisioner** ホストの、とくに **baremetal** ネットワークブリッジについて接続を確認します。

- **Permission denied (publickey,password,keyboard-interactive)** が出される。ブートストラップ仮想マシンへのアクセスを試行すると、**Permission denied** エラーが発生する可能性があります。仮想マシンへのログインを試行するユーザーの SSH キーが **install-config.yaml** ファイル内で設定されていることを確認します。

16.6.3.1. ブートストラップ仮想マシンがクラスターノードを起動できない

デプロイメント時に、ブートストラップ仮想マシンがクラスターノードの起動に失敗する可能性があり、これにより、仮想マシンがノードに RHCOS イメージをプロビジョニングできなくなります。このシナリオは、以下の原因で発生する可能性があります。

- **install-config.yaml** ファイルに関連する問題。

- ベアメタルネットワークを使用してアウトオブバンド (out-of-band) ネットワークアクセスに関する問題

この問題を確認するには、**ironic** に関連する 3 つのコンテナーを使用できます。

- **ironic**

- **ironic-inspector**

手順

1. ブートストラップ仮想マシンにログインします。

```
$ ssh core@172.22.0.2
```

2. コンテナーログを確認するには、以下を実行します。

```
[core@localhost ~]$ sudo podman logs -f <container_name>
```

<container_name> を **ironic** または **ironic-inspector** のいずれかに置き換えます。コントロールプレーンノードが PXE から起動しないという問題が発生した場合は、**ironic** Pod を確認してください。**ironic** Pod は、IPMI 経由でノードにログインしようとするため、クラスターノードを起動しようとする試みに関する情報を含んでいます。

考えられる理由

クラスターノードは、デプロイメントの開始時に **ON** 状態にある可能性があります。

解決策

IPMI でのインストールを開始する前に、OpenShift Container Platform クラスターノードの電源をオフにします。

```
$ ipmitool -I lanplus -U root -P <password> -H <out_of_band_ip> power off
```

16.6.3.2. ログの検査

RHCOS イメージのダウンロードまたはアクセスに問題が発生した場合には、最初に **install-config.yaml** 設定ファイルで URL が正しいことを確認します。

RHCOS イメージをホストする内部 Web サーバーの例

```
bootstrapOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-qemu.<architecture>.qcow2.gz?sha256=9d999f55ff1d44f7ed7c106508e5deecd04dc3c06095d34d36bf1cd127837e0c
clusterOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-openstack.<architecture>.qcow2.gz?sha256=a1bda656fa0892f7b936fdc6b6a6086bddaed5dafacedcd7a1e811abb78fe3b0
```

coreos-downloader コンテナーは、Web サーバーまたは外部の quay.io レジストリー (**install-config.yaml** 設定ファイルで指定されている方) からリソースをダウンロードします。**coreos-downloader** コンテナーが稼働していることを確認し、必要に応じて、そのログを調べます。

手順

1. ブートストラップ仮想マシンにログインします。

   ```
   $ ssh core@172.22.0.2
   ```

2. 次のコマンドを実行して、ブートストラップ VM 内の **coreos-downloader** コンテナーのステータスを確認します。

   ```
   [core@localhost ~]$ sudo podman logs -f coreos-downloader
   ```

   ブートストラップ仮想マシンがイメージへの URL にアクセスできない場合、**curl** コマンドを使用して、仮想マシンがイメージにアクセスできることを確認します。

3. すべてのコンテナーがデプロイメントフェーズで起動されているかどうかを示す **bootkube** ログを検査するには、以下を実行します。

   ```
   [core@localhost ~]$ journalctl -xe
   ```

   ```
   [core@localhost ~]$ journalctl -b -f -u bootkube.service
   ```

4. **dnsmasq**、**mariadb**、**httpd**、および **ironic** を含むすべての Pod が実行中であることを確認します。

   ```
   [core@localhost ~]$ sudo podman ps
   ```

5. Pod に問題がある場合には、問題のあるコンテナーのログを確認します。**ironic** サービスのログを確認するには、次のコマンドを実行します。

   ```
   [core@localhost ~]$ sudo podman logs ironic
   ```

16.6.4. 利用できない Kubernetes API の調査

Kubernetes API が利用できない場合は、コントロールプレーンノードをチェックして、正しいコンポーネントが実行されていることを確認します。また、ホスト名の解決も確認してください。

手順

1. 次のコマンドを実行して、各コントロールプレーンノードで **etcd** が実行されていることを確認します。

```
$ sudo crictl logs $(sudo crictl ps --pod=$(sudo crictl pods --name=etcd-member --quiet) --quiet)
```

2. 直前のコマンドが失敗する場合、以下のコマンドを実行して Kubelet が **etcd** Pod を作成していることを確認します。

```
$ sudo crictl pods --name=etcd-member
```

Pod がない場合は、**etcd** を調査します。

3. 次のコマンドを使用して、クラスターノードに **localhost.localdomain** だけでなく完全修飾ドメイン名があることを確認します。

```
$ hostname
```

ホスト名が設定されていない場合、正しいホスト名を設定します。以下に例を示します。

```
$ sudo hostnamectl set-hostname <hostname>
```

4. **dig** コマンドを使用して、各ノードの DNS サーバーでの名前解決が正しいことを確認します。

```
$ dig api.<cluster_name>.example.com
```

```
; <<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <<>> api.<cluster_name>.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37551
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 866929d2f8e8563582af23f05ec44203d313e50948d43f60 (good)
;; QUESTION SECTION:
;api.<cluster_name>.example.com. IN A

;; ANSWER SECTION:
api.<cluster_name>.example.com. 10800 IN A 10.19.13.86

;; AUTHORITY SECTION:
<cluster_name>.example.com. 10800 IN NS <cluster_name>.example.com.

;; ADDITIONAL SECTION:
<cluster_name>.example.com. 10800 IN A 10.19.14.247

;; Query time: 0 msec
;; SERVER: 10.19.14.247#53(10.19.14.247)
;; WHEN: Tue May 19 20:30:59 UTC 2020
;; MSG SIZE  rcvd: 140
```

前述の例の出力は、**api.<cluster_name>.example.com** VIP の適切な IP アドレスが **10.19.13.86** であることを示しています。この IP アドレスは **baremetal** 上にある必要があります。

16.6.5. クラスターの初期化失敗のトラブルシューティング

インストールプログラムは、Cluster Version Operator を使用して、OpenShift Container Platform クラスターのすべてのコンポーネントを作成します。インストールプログラムがクラスターの初期化に失敗した場合は、**ClusterVersion** オブジェクトと **ClusterOperator** オブジェクトから最も重要な情報を取得できます。

手順

1. 次のコマンドを実行して **ClusterVersion** オブジェクトを検査します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get clusterversion -o yaml
```

出力例

```
apiVersion: config.openshift.io/v1
kind: ClusterVersion
metadata:
  creationTimestamp: 2019-02-27T22:24:21Z
  generation: 1
  name: version
  resourceVersion: "19927"
  selfLink: /apis/config.openshift.io/v1/clusterversions/version
  uid: 6e0f4cf8-3ade-11e9-9034-0a923b47ded4
spec:
  channel: stable-4.1
  clusterID: 5ec312f9-f729-429d-a454-61d4906896ca
status:
  availableUpdates: null
  conditions:
  - lastTransitionTime: 2019-02-27T22:50:30Z
    message: Done applying 4.1.1
    status: "True"
    type: Available
  - lastTransitionTime: 2019-02-27T22:50:30Z
    status: "False"
    type: Failing
  - lastTransitionTime: 2019-02-27T22:50:30Z
    message: Cluster version is 4.1.1
    status: "False"
    type: Progressing
  - lastTransitionTime: 2019-02-27T22:24:31Z
    message: 'Unable to retrieve available updates: unknown version 4.1.1
    reason: RemoteFailed
    status: "False"
    type: RetrievedUpdates
  desired:
    image: registry.svc.ci.openshift.org/openshift/origin-
release@sha256:91e6f754975963e7db1a9958075eb609ad226968623939d262d1cf45e9dbc3
9a
    version: 4.1.1
```

```
  history:
  - completionTime: 2019-02-27T22:50:30Z
    image: registry.svc.ci.openshift.org/openshift/origin-
release@sha256:91e6f754975963e7db1a9958075eb609ad226968623939d262d1cf45e9dbc3
9a
    startedTime: 2019-02-27T22:24:31Z
    state: Completed
    version: 4.1.1
  observedGeneration: 1
  versionHash: Wa7as_ik1qE=
```

2. 次のコマンドを実行して状態を表示します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get clusterversion version \
    -o=jsonpath='{range .status.conditions[*]}{.type}{" "}{.status}{" "}{.message}{"\n"}{end}'
```

最も重要な状態は、**Failing**、**Available**、**Progressing** です。

出力例

```
Available True Done applying 4.1.1
Failing False
Progressing False Cluster version is 4.0.0-0.alpha-2019-02-26-194020
RetrievedUpdates False Unable to retrieve available updates: unknown version 4.1.1
```

3. 次のコマンドを実行して **ClusterOperator** オブジェクトを検査します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get clusteroperator
```

このコマンドは、クラスター Operators のステータスを返します。

出力例

```
NAME                                 VERSION   AVAILABLE   PROGRESSING   FAILING   SINCE
cluster-baremetal-operator                     True        False         False     17m
cluster-autoscaler                       True        False         False     17m
cluster-storage-operator                   True        False         False     10m
console                                  True        False         False     7m21s
dns                                     True        False         False     31m
image-registry                           True        False         False     9m58s
ingress                                 True        False         False     10m
kube-apiserver                           True        False         False     28m
kube-controller-manager                        True        False         False     21m
kube-scheduler                           True        False         False     25m
machine-api                             True        False         False     17m
machine-config                           True        False         False     17m
marketplace-operator                           True        False         False     10m
monitoring                              True        False         False     8m23s
network                                 True        False         False     13m
node-tuning                              True        False         False     11m
openshift-apiserver                            True        False         False     15m
openshift-authentication                       True        False         False     20m
openshift-cloud-credential-operator            True        False         False     18m
openshift-controller-manager                   True        False         False     10m
```

```
openshift-samples                          True       False       False     8m42s
operator-lifecycle-manager                 True       False       False     17m
service-ca                                 True       False       False     30m
```

4. 次のコマンドを実行して、個々のクラスター Operator を検査します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get clusteroperator <operator> -oyaml
```
(1)

(1) **<operator>** は、クラスター Operator の名前に置き換えます。このコマンドは、クラスター Operator のステータスが **Available** にならない理由、または **Failed** になる理由を特定するために使用できます。

出力例

```
apiVersion: config.openshift.io/v1
kind: ClusterOperator
metadata:
  creationTimestamp: 2019-02-27T22:47:04Z
  generation: 1
  name: monitoring
  resourceVersion: "24677"
  selfLink: /apis/config.openshift.io/v1/clusteroperators/monitoring
  uid: 9a6a5ef9-3ae1-11e9-bad4-0a97b6ba9358
spec: {}
status:
  conditions:
  - lastTransitionTime: 2019-02-27T22:49:10Z
    message: Successfully rolled out the stack.
    status: "True"
    type: Available
  - lastTransitionTime: 2019-02-27T22:49:10Z
    status: "False"
    type: Progressing
  - lastTransitionTime: 2019-02-27T22:49:10Z
    status: "False"
    type: Failing
  extension: null
  relatedObjects: null
  version: ""
```

5. クラスター Operator のステータス状態を取得するには、次のコマンドを実行します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get clusteroperator <operator> \
    -o=jsonpath='{range .status.conditions[*]}{.type}{" "}{.status}{" "}{.message}{"\n"}{end}'
```

<Operator> は、上記のいずれかの Operator の名前に置き換えます。

出力例

```
Available True Successfully rolled out the stack
Progressing False
Failing False
```

6. クラスター Operator が所有するオブジェクトのリストを取得するには、次のコマンドを実行します。

```
oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get clusteroperator kube-apiserver \
  -o=jsonpath='{.status.relatedObjects}'
```

出力例

```
[map[resource:kubeapiservers group:operator.openshift.io name:cluster] map[group:
name:openshift-config resource:namespaces] map[group: name:openshift-config-managed
resource:namespaces] map[group: name:openshift-kube-apiserver-operator
resource:namespaces] map[group: name:openshift-kube-apiserver resource:namespaces]]
```

16.6.6. コンソール URL の取得に失敗した場合のトラブルシューティング

インストールプログラムは、**openshift-console** namespace 内の **[route][route-object]** を使用して、OpenShift Container Platform コンソールの URL を取得します。インストールプログラムがコンソールの URL の取得に失敗した場合は、次の手順に従います。

手順

1. 次のコマンドを実行して、コンソールルーターが **Available** 状態か **Failing** 状態かを確認します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get clusteroperator console -oyaml
```

```
apiVersion: config.openshift.io/v1
kind: ClusterOperator
metadata:
  creationTimestamp: 2019-02-27T22:46:57Z
  generation: 1
  name: console
  resourceVersion: "19682"
  selfLink: /apis/config.openshift.io/v1/clusteroperators/console
  uid: 960364aa-3ae1-11e9-bad4-0a97b6ba9358
spec: {}
status:
  conditions:
  - lastTransitionTime: 2019-02-27T22:46:58Z
    status: "False"
    type: Failing
  - lastTransitionTime: 2019-02-27T22:50:12Z
    status: "False"
    type: Progressing
  - lastTransitionTime: 2019-02-27T22:50:12Z
    status: "True"
    type: Available
  - lastTransitionTime: 2019-02-27T22:46:57Z
    status: "True"
    type: Upgradeable
  extension: null
  relatedObjects:
  - group: operator.openshift.io
    name: cluster
```

```
    resource: consoles
  - group: config.openshift.io
    name: cluster
    resource: consoles
  - group: oauth.openshift.io
    name: console
    resource: oauthclients
  - group: ""
    name: openshift-console-operator
    resource: namespaces
  - group: ""
    name: openshift-console
    resource: namespaces
  versions: null
```

2. 次のコマンドを実行して、コンソール URL を手動で取得します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get route console -n openshift-console \
    -o=jsonpath='{.spec.host}' console-openshift-console.apps.adahiya-
1.devcluster.openshift.com
```

16.6.7. kubeconfig に Ingress 証明書を追加できない場合のトラブルシューティング

インストールプログラムは、**${INSTALL_DIR}/auth/kubeconfig** 内の信頼できるクライアント認証局のリストにデフォルトの Ingress 証明書を追加します。インストールプログラムが Ingress 証明書を **kubeconfig** ファイルに追加できない場合は、クラスターから証明書を取得して追加できます。

手順

1. 次のコマンドを使用して、クラスターから証明書を取得します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig get configmaps default-ingress-cert \
    -n openshift-config-managed -o=jsonpath='{.data.ca-bundle\.crt}'
```

```
-----BEGIN CERTIFICATE-----
MIIC/TCCAeWgAwIBAgIBATANBgkqhkiG9w0BAQsFADAuMSwwKgYDVQQDDCNjbHVz
dGVyLWluZ3Jlc3Mtb3BlcmF0b3JAMTU1MTMwNzU4OTAeFw0xOTAyMjcyMjQ2Mjha
Fw0yMTAyMjYyMjQ2MjlaMC4xLDAqBgNVBAMMI2NsdXN0ZXItaW5ncmVzcy1vcGVy
YXRvckAxNTUxMzA3NTg5MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
uCA4fQ+2YXoXSUL4h/mcvJfrgpBfKBW5hfB8NcgXeCYiQPnCKblH1sEQnI3VC5Pk
2OfNCF3PUlfm4i8CHC95a7nCkRjmJNg1gVrWCvS/ohLgnO0BvszSiRLxIpuo3C4S
EVqqvxVaIHcbdAXWgZLQoYZXV7RMz8yZjl5CfhDaaItyBFj3GtIJkXgUwp/5sUfI
LDXW8MM6AXfuG+kweLdLCMm3g8WLLfLBLvVBKB+4IhIH7Il0buOz04RKhnYN+Ebw
tcvFi55vwuUCWMnGhWHGEQ8sWm/wLnNlOwsUz7S1/sW8nj87GFHzgkaVM9EOnoNI
gKhMBK9ItNzjrP6dgiKBCQIDAQABoyYwJDAOBgNVHQ8BAf8EBAMCAqQwEgYDVR0T
AQH/BAgwBgEB/wIBADANBgkqhkiG9w0BAQsFAAOCAQEAq+vi0sFKudaZ9aUQMMha
CeWx9CZvZBblnAWT/61UdpZKpFi4eJ2d33lGcfKwHOi2NP/iSKQBebfG0iNLVVPz
vwLbSG1i9R9GLdAbnHpPT9UG6fLaDIoKpnKiBfGENfxeiq5vTln2bAgivxrVlyiq
+MdDXFAWb6V4u2xh6RChI7akNsS3oU9PZ9YOs5e8vJp2YAEphht05X0swA+X8V8T
C278FFifpo0h3Q0Dbv8Rfn4UpBEtN4KkLeS+JeT+0o2XOsFZp7Uhr9yFIodRsnNo
H/Uwmab28ocNrGNiEVaVH6eTTQeeZuOdoQzUbClElpVmkrNGY0M42K0PvOQ/e7+y
AQ==
-----END CERTIFICATE-----
```

2. **${INSTALL_DIR}/auth/kubeconfig** ファイルの **client-certificate-authority-data** フィールドに証明書を追加します。

16.6.8. クラスターノードへの SSH アクセスのトラブルシューティング

セキュリティーを強化するために、デフォルトではクラスターの外部からクラスターに SSH 接続することは禁止されています。ただし、プロビジョナーノードからコントロールプレーンノードとワーカーノードにアクセスすることはできます。プロビジョナーノードからクラスターノードに SSH 接続できない場合、クラスターノードがブートストラップ仮想マシンを待機している可能性があります。コントロールプレーンノードはブートストラップ仮想マシンからブート設定を取得します。コントロールプレーンノードはブート設定を取得しないと、正常に起動できません。

手順

1. ノードに物理的にアクセスできる場合は、コンソールの出力をチェックして、正常に起動したかどうかを確認します。ノードがまだブート設定を取得中の場合は、ブートストラップ仮想マシンに問題がある可能性があります。

2. **install-config.yaml** ファイルで **sshKey: '<ssh_pub_key>'** が設定されていることを確認します。**<ssh_pub_key>** は、プロビジョナーノード上の **kni** ユーザーの公開鍵です。

16.6.9. クラスターノードが PXE ブートしない

OpenShift Container Platform クラスターノードが PXE ブートしない場合、PXE ブートしないクラスターノードで以下のチェックを実行します。この手順は、**provisioning** ネットワークなしで OpenShift Container Platform クラスターをインストールする場合には適用されません。

手順

1. **provisioning** ネットワークへのネットワークの接続を確認します。

2. PXE が **provisioning** ネットワークの NIC で有効にされており、PXE がその他のすべての NIC について無効にされていることを確認します。

3. **install-config.yaml** 設定ファイルに、**provisioning** ネットワークに接続されている NIC の **rootDeviceHints** パラメーターとブート MAC アドレスが含まれていることを確認します。以下に例を示します。

   コントロールプレーンノードの設定

   ```
   bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
   ```

   ワーカーノード設定

   ```
   bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
   ```

16.6.10. インストールしてもワーカーノードが作成されない

インストールプログラムはワーカーノードを直接プロビジョニングしません。代わりに、Machine API Operator が、サポートされているプラットフォーム上でノードをスケールアップおよびスケールダウンします。クラスターのインターネット接続の速度によって異なりますが、15 - 20 分経ってもワーカーノードが作成されない場合は、Machine API Operator を調査してください。

手順

1. 次のコマンドを実行して、Machine APIOperator を確認します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig \
  --namespace=openshift-machine-api get deployments
```

ご使用の環境で **${INSTALL_DIR}** が設定されていない場合は、値をインストールディレクトリーの名前に置き換えます。

出力例

```
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
cluster-autoscaler-operator   1/1     1            1           86m
cluster-baremetal-operator    1/1     1            1           86m
machine-api-controllers       1/1     1            1           85m
machine-api-operator          1/1     1            1           86m
```

2. 次のコマンドを実行して、マシンコントローラーのログを確認します。

```
$ oc --kubeconfig=${INSTALL_DIR}/auth/kubeconfig \
    --namespace=openshift-machine-api logs deployments/machine-api-controllers \
    --container=machine-controller
```

16.6.11. Cluster Network Operator のトラブルシューティング

Cluster Network Operator は、ネットワークコンポーネントのデプロイを担当します。この Operator は、コントロールプレーンノードが起動した後、インストールプログラムがブートストラップコントロールプレーンを削除する前に、インストールプロセスの初期段階で実行されます。この Operator に問題がある場合、インストールプログラムに問題がある可能性があります。

手順

1. 次のコマンドを実行して、ネットワーク設定が存在することを確認します。

```
$ oc get network -o yaml cluster
```

存在しない場合は、インストールプログラムによって作成されていません。理由を確認するには、次のコマンドを実行します。

```
$ openshift-install create manifests
```

マニフェストを確認して、インストールプログラムがネットワーク設定を作成しなかった理由を特定します。

2. 次のコマンドを入力して、ネットワークが実行中であることを確認します。

```
$ oc get po -n openshift-network-operator
```

16.6.12. BMC を使用して、新しいベアメタルホストを検出できない

場合によっては、リモート仮想メディア共有をマウントできないため、インストールプログラムが新しいベアメタルホストを検出できず、エラーが発生することがあります。

以下に例を示します。

```
ProvisioningError 51s metal3-baremetal-controller Image provisioning failed: Deploy step
deploy.deploy failed with BadRequestError: HTTP POST
https://<bmc_address>/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia/CD/Actions/VirtualMedia.
InsertMedia
returned code 400.
Base.1.8.GeneralError: A general error has occurred. See ExtendedInfo for more information
Extended information: [
  {
    "Message": "Unable to mount remote share https://<ironic_address>/redfish/boot-<uuid>.iso.",
    "MessageArgs": [
      "https://<ironic_address>/redfish/boot-<uuid>.iso"
    ],
    "MessageArgs@odata.count": 1,
    "MessageId": "IDRAC.2.5.RAC0720",
    "RelatedProperties": [
      "#/Image"
    ],
    "RelatedProperties@odata.count": 1,
    "Resolution": "Retry the operation.",
    "Severity": "Informational"
  }
].
```

この状況で、認証局が不明な仮想メディアを使用している場合は、不明な認証局を信頼するように、ベースボード管理コントローラー (BMC) のリモートファイル共有設定を行って、このエラーを回避できます。

注記

この解決策は、Dell iDRAC 9 およびファームウェアバージョン 5.10.50 を搭載した OpenShift Container Platform 4.11 でテストされました。

16.6.13. クラスターに参加できないワーカーノードのトラブルシューティング

インストーラーでプロビジョニングされるクラスターは、**api-int.<cluster_name>.<base_domain>** URL の DNS エントリーが含まれる DNS サーバーと共にデプロイされます。クラスター内のノードが外部またはアップストリーム DNS サーバーを使用して **api-int.<cluster_name>.<base_domain>** URL を解決し、そのようなエントリーがない場合、ワーカーノードはクラスターに参加できない可能性があります。クラスター内のすべてのノードがドメイン名を解決できることを確認します。

手順

1. DNS A/AAAA または CNAME レコードを追加して、API ロードバランサーを内部的に識別します。たとえば、dnsmasq を使用する場合は、**dnsmasq.conf** 設定ファイルを変更します。

```
$ sudo nano /etc/dnsmasq.conf
```

```
address=/api-int.<cluster_name>.<base_domain>/<IP_address>
address=/api-int.mycluster.example.com/192.168.1.10
address=/api-int.mycluster.example.com/2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

2. DNS PTR レコードを追加して、API ロードバランサーを内部的に識別します。たとえば、dnsmasq を使用する場合は、**dnsmasq.conf** 設定ファイルを変更します。

```
$ sudo nano /etc/dnsmasq.conf
```

```
ptr-record=<IP_address>.in-addr.arpa,api-int.<cluster_name>.<base_domain>
ptr-record=10.1.168.192.in-addr.arpa,api-int.mycluster.example.com
```

3. DNS サーバーを再起動します。たとえば、dnsmasq を使用する場合は、以下のコマンドを実行します。

```
$ sudo systemctl restart dnsmasq
```

これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。

16.6.14. 以前のインストールのクリーンアップ

以前のデプロイメントに失敗した場合、OpenShift Container Platform のデプロイを再試行する前に、失敗した試行からアーティファクトを削除します。

手順

1. OpenShift Container Platform クラスターをインストールする前に、すべてのベアメタルノードの電源をオフにします。

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management_server_ip> power off
```

2. 以前に試行したデプロイメントにより古いブートストラップリソースが残っている場合は、これらをすべて削除します。

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

16.6.15. レジストリーの作成に関する問題

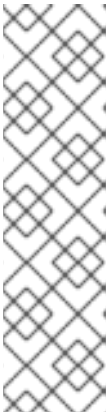
非接続レジストリーの作成時に、レジストリーのミラーリングを試行する際に User Not Authorized エラーが発生する場合があります。このエラーは、新規の認証を既存の **pull-secret.txt** ファイルに追加できない場合に生じる可能性があります。

手順

1. 認証が正常に行われていることを確認します。

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.json
  --from=$UPSTREAM_REPO \
```

```
--to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
--to=$LOCAL_REG/$LOCAL_REPO
```



注記

インストールイメージのミラーリングに使用される変数の出力例:

```
UPSTREAM_REPO=${RELEASE_IMAGE}
LOCAL_REG=<registry_FQDN>:<registry_port>
LOCAL_REPO='ocp4/openshift4'
```

RELEASE_IMAGE および **VERSION** の値は、**OpenShift インストールの環境のセットアップ**セクションの **OpenShift Installer の取得** の手順で設定されています。

2. レジストリーのミラーリング後に、非接続環境でこれにアクセスできることを確認します。

```
$ curl -k -u <user>:<password> https://registry.example.com:<registry_port>/v2/_catalog
{"repositories":["<Repo_Name>"]}
```

16.6.16. その他の問題点

16.6.16.1. runtime network not ready エラーへの対応

クラスターのデプロイメント後に、以下のエラーが発生する可能性があります。

```
`runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady message:Network
plugin returns error: Missing CNI default network`
```

Cluster Network Operator は、インストールプログラムによって作成される特別なオブジェクトに応じてネットワークコンポーネントをデプロイします。これは、コントロールプレーン (マスター) ノードが起動した後、ブートストラップコントロールプレーンが停止する前にインストールプロセスの初期段階で実行されます。これは、コントロールプレーン (マスター) ノードの起動の長い遅延や **apiserver** の通信の問題など、より判別しにくいインストールプログラムの問題を示している可能性があります。

手順

1. **openshift-network-operator** namespace の Pod を検査します。

```
$ oc get all -n openshift-network-operator
```

```
NAME                                    READY STATUS            RESTARTS   AGE
pod/network-operator-69dfd7b577-bg89v   0/1   ContainerCreating 0          149m
```

2. **provisioner** ノードで、ネットワーク設定が存在することを判別します。

```
$ kubectl get network.config.openshift.io cluster -oyaml
```

```
apiVersion: config.openshift.io/v1
kind: Network
metadata:
```

```
  name: cluster
spec:
  serviceNetwork:
  - 172.30.0.0/16
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OVNKubernetes
```

存在しない場合は、インストールプログラムによって作成されていません。インストールプログラムによって作成されなかった理由を確認するには、次のコマンドを実行します。

```
$ openshift-install create manifests
```

3. **network-operator** が実行されていることを確認します。

```
$ kubectl -n openshift-network-operator get pods
```

4. ログを取得します。

```
$ kubectl -n openshift-network-operator logs -l "name=network-operator"
```

3 つ以上のコントロールプレーンノードを持つ高可用性クラスターでは、Operator がリーダー選択を実行し、他のすべての Operator がスリープ状態になります。詳細は、Troubleshooting を参照してください。

16.6.16.2. "No disk found with matching rootDeviceHints" エラーメッセージへの対処

クラスターをデプロイした後、次のエラーメッセージが表示される場合があります。

```
No disk found with matching rootDeviceHints
```

No disk found with matching rootDeviceHints エラーメッセージに対処するための一時的な回避策は、**rootDeviceHints** を **minSizeGigabytes: 300** に変更することです。

rootDeviceHints 設定を変更した後、CoreOS を起動し、次のコマンドを使用してディスク情報を確認します。

```
$ udevadm info /dev/sda
```

DL360 Gen 10 サーバーを使用している場合は、**/dev/sda** デバイス名が割り当てられている SD カードスロットがあることに注意してください。サーバーに SD カードが存在しない場合は、競合が発生する可能性があります。サーバーの BIOS 設定で SD カードスロットが無効になっていることを確認してください。

minSizeGigabytes の回避策が要件を満たしていない場合は、**rootDeviceHints** を **/dev/sda** に戻さないといけない場合があります。この変更により、Ironic イメージが正常に起動できるようになります。

この問題を解決する別の方法は、ディスクのシリアル ID を使用することです。ただし、シリアル ID を見つけるのは困難な場合があり、設定ファイルが読みにくくなる可能性があることに注意してください。このパスを選択する場合は、前に説明したコマンドを使用してシリアル ID を収集し、それを設定に組み込んでください。

16.6.16.3. クラスターノードが DHCP 経由で正しい IPv6 アドレスを取得しない

クラスターノードが DHCP 経由で正しい IPv6 アドレスを取得しない場合は、以下の点を確認してください。

1. 予約された IPv6 アドレスが DHCP 範囲外にあることを確認します。
2. DHCP サーバーの IP アドレス予約では、予約で正しい DUID (DHCP 固有識別子) が指定されていることを確認します。以下に例を示します。

```
# This is a dnsmasq dhcp reservation, 'id:00:03:00:01' is the client id and '18:db:f2:8c:d5:9f' is the MAC Address for the NIC
id:00:03:00:01:18:db:f2:8c:d5:9f,openshift-master-1,[2620:52:0:1302::6]
```

3. Route Announcement が機能していることを確認します。
4. DHCP サーバーが、IP アドレス範囲を提供する必要なインターフェイスでリッスンしていることを確認します。

16.6.16.4. クラスターノードが DHCP 経由で正しいホスト名を取得しない

IPv6 のデプロイメント時に、クラスターノードは DHCP でホスト名を取得する必要があります。**NetworkManager** はホスト名をすぐに割り当てない場合があります。コントロールプレーン (マスター) ノードは、以下のようなエラーを報告する可能性があります。

```
Failed Units: 2
  NetworkManager-wait-online.service
  nodeip-configuration.service
```

このエラーは、最初に DHCP サーバーからホスト名を受信せずにクラスターノードが起動する可能性があることを示しています。これにより、**kubelet** が **localhost.localdomain** ホスト名で起動します。エラーに対処するには、ノードによるホスト名の更新を強制します。

手順

1. **hostname** を取得します。

```
[core@master-X ~]$ hostname
```

   ホスト名が **localhost** の場合は、以下の手順に進みます。

注記

X はコントロールプレーンノード番号です。

2. クラスターノードによる DHCP リースの更新を強制します。

```
[core@master-X ~]$ sudo nmcli con up "<bare_metal_nic>"
```

   <bare_metal_nic> を、**baremetal** ネットワークに対応する有線接続に置き換えます。

3. **hostname** を再度確認します。

```
[core@master-X ~]$ hostname
```

4. ホスト名が **localhost.localdomain** の場合は、 **NetworkManager** を再起動します。

```
[core@master-X ~]$ sudo systemctl restart NetworkManager
```

5. ホスト名がまだ **localhost.localdomain** の場合は、数分待機してから再度確認します。ホスト名が **localhost.localdomain** のままの場合は、直前の手順を繰り返します。

6. **nodeip-configuration** サービスを再起動します。

```
[core@master-X ~]$ sudo systemctl restart nodeip-configuration.service
```

このサービスは、正しいホスト名の参照で **kubelet** サービスを再設定します。

7. kubelet が直前の手順で変更された後にユニットファイル定義を再読み込みします。

```
[core@master-X ~]$ sudo systemctl daemon-reload
```

8. **kubelet** サービスを再起動します。

```
[core@master-X ~]$ sudo systemctl restart kubelet.service
```

9. **kubelet** が正しいホスト名で起動されていることを確認します。

```
[core@master-X ~]$ sudo journalctl -fu kubelet.service
```

再起動時など、クラスターの稼働後にクラスターノードが正しいホスト名を取得しない場合、クラスターの **csr** は保留中になります。**csr** は承認 **しません**。承認すると、他の問題が生じる可能性があります。

csr の対応

1. クラスターで CSR を取得します。

```
$ oc get csr
```

2. 保留中の **csr** に **Subject Name: localhost.localdomain** が含まれているかどうかを確認します。

```
$ oc get csr <pending_csr> -o jsonpath='{.spec.request}' | base64 --decode | openssl req -noout -text
```

3. **Subject Name: localhost.localdomain** が含まれる **csr** を削除します。

```
$ oc delete csr <wrong_csr>
```

16.6.16.5. ルートがエンドポイントに到達しない

インストールプロセス時に、VRRP (Virtual Router Redundancy Protocol) の競合が発生する可能性があります。この競合は、特定のクラスター名を使用してクラスターデプロイメントの一部であった、以前に使用された OpenShift Container Platform ノードが依然として実行中であるものの、同じクラスター名を使用した現在の OpenShift Container Platform クラスターデプロイメントの一部ではない場合に発

生する可能性があります。たとえば、クラスターはクラスター名 **openshift** を使用してデプロイされ、3 つのコントロールプレーン (マスター) ノードと 3 つのワーカーノードをデプロイします。後に、別のインストールで同じクラスター名 **openshift** が使用されますが、この再デプロイメントは 3 つのコントロールプレーン (マスター) ノードのみをインストールし、以前のデプロイメントの 3 つのワーカーノードを **ON** 状態のままにします。これにより、VRID (Virtual Router Identifier) の競合が発生し、VRRP が競合する可能性があります。

1. ルートを取得します。

```
$ oc get route oauth-openshift
```

2. サービスエンドポイントを確認します。

```
$ oc get svc oauth-openshift
```

```
NAME              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
oauth-openshift   ClusterIP   172.30.19.162   <none>        443/TCP   59m
```

3. コントロールプレーン (マスター) ノードからサービスへのアクセスを試行します。

```
[core@master0 ~]$ curl -k https://172.30.19.162
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {
  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
```

4. **provisioner** ノードからの **authentication-operator** エラーを特定します。

```
$ oc logs deployment/authentication-operator -n openshift-authentication-operator
```

```
Event(v1.ObjectReference{Kind:"Deployment", Namespace:"openshift-authentication-
operator", Name:"authentication-operator", UID:"225c5bd5-b368-439b-9155-5fd3c0459d98",
APIVersion:"apps/v1", ResourceVersion:"", FieldPath:""}): type: 'Normal' reason:
'OperatorStatusChanged' Status for clusteroperator/authentication changed: Degraded
message changed from "IngressStateEndpointsDegraded: All 2 endpoints for oauth-server
are reporting"
```

解決策

1. すべてのデプロイメントのクラスター名が一意であり、競合が発生しないことを確認します。

2. 同じクラスター名を使用するクラスターデプロイメントの一部ではない不正なノードをすべてオフにします。そうしないと、OpenShift Container Platform クラスターの認証 Pod が正常に起動されなくなる可能性があります。

16.6.16.6. 初回起動時の Ignition の失敗

初回起動時に、Ignition 設定が失敗する可能性があります。

手順

1. Ignition 設定が失敗したノードに接続します。

```
Failed Units: 1
  machine-config-daemon-firstboot.service
```

2. **machine-config-daemon-firstboot** サービスを再起動します。

```
[core@worker-X ~]$ sudo systemctl restart machine-config-daemon-firstboot.service
```

16.6.16.7. NTP が同期しない

OpenShift Container Platform クラスターのデプロイメントは、クラスターノード間の NTP の同期クロックによって異なります。同期クロックがない場合、時間の差が 2 秒を超えるとクロックのドリフトによりデプロイメントが失敗する可能性があります。

手順

1. クラスターノードの **AGE** の差異の有無を確認します。以下に例を示します。

```
$ oc get nodes
```

```
NAME                         STATUS   ROLES    AGE   VERSION
master-0.cloud.example.com   Ready    master   145m   v1.29.4
master-1.cloud.example.com   Ready    master   135m   v1.29.4
master-2.cloud.example.com   Ready    master   145m   v1.29.4
worker-2.cloud.example.com   Ready    worker   100m   v1.29.4
```

2. クロックのドリフトによる一貫性のないタイミングの遅延について確認します。以下に例を示します。

```
$ oc get bmh -n openshift-machine-api
```

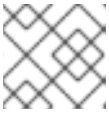
```
master-1    error registering master-1  ipmi://<out_of_band_ip>
```

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 18:20:02 UTC
           Universal time: Tue 2020-03-10 18:20:02 UTC
                 RTC time: Tue 2020-03-10 18:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no
```

既存のクラスターでのクロックドリフトへの対応

1. ノードに配信される **chrony.conf** ファイルの内容を含む Butane 設定ファイルを作成します。以下の例で、**99-master-chrony.bu** を作成して、ファイルをコントロールプレーンノードに追加します。ワーカーノードのファイルを変更するか、ワーカーロールに対してこの手順を繰り返すことができます。



注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-master-chrony
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        server <NTP_server> iburst (1)
        stratumweight 0
        driftfile /var/lib/chrony/drift
        rtcsync
        makestep 10 3
        bindcmdaddress 127.0.0.1
        bindcmdaddress ::1
        keyfile /etc/chrony.keys
        commandkey 1
        generatecommandkey
        noclientlog
        logchange 0.5
        logdir /var/log/chrony
```

(1) **<NTP_server>** を NTP サーバーの IP アドレスに置き換えます。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-master-chrony.yaml**) を生成します。

```
$ butane 99-master-chrony.bu -o 99-master-chrony.yaml
```

3. **MachineConfig** オブジェクトファイルを適用します。

```
$ oc apply -f 99-master-chrony.yaml
```

4. **System clock synchronized** の値が **yes** であることを確認します。

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 19:10:02 UTC
```

```
               Universal time: Tue 2020-03-10 19:10:02 UTC
                     RTC time: Tue 2020-03-10 19:36:53
                    Time zone: UTC (UTC, +0000)
System clock synchronized: yes
                  NTP service: active
              RTC in local TZ: no
```

デプロイメントの前にクロック同期を設定するには、マニフェストファイルを生成し、このファイルを **openshift** ディレクトリーに追加します。以下に例を示します。

```
$ cp chrony-masters.yaml ~/clusterconfigs/openshift/99_masters-chrony-configuration.yaml
```

クラスターの作成を継続します。

16.6.17. インストールの確認

インストール後、インストールプログラムによってノードと Pod が正常にデプロイされたことを確認します。

手順

1. OpenShift Container Platform クラスターノードが適切にインストールされると、以下の **Ready** 状態が **STATUS** 列に表示されます。

```
$ oc get nodes
```

```
NAME                 STATUS   ROLES          AGE  VERSION
master-0.example.com   Ready    master,worker   4h   v1.29.4
master-1.example.com   Ready    master,worker   4h   v1.29.4
master-2.example.com   Ready    master,worker   4h   v1.29.4
```

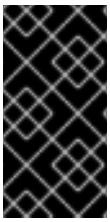
2. インストールプログラムによってすべての Pod が正常にデプロイされたことを確認します。以下のコマンドは、実行中の Pod、または出力の一部として完了した Pod を削除します。

```
$ oc get pods --all-namespaces | grep -iv running | grep -iv complete
```

第17章 IBM CLOUD BARE METAL (CLASSIC) のインストール

17.1. 前提条件

インストーラーによってプロビジョニングされたインストールを使用して、OpenShift Container Platform を IBM Cloud® Bare Metal (Classic) ノードにインストールできます。本書では、IBM Cloud® ノードに OpenShift Container Platform をインストールする際の前提条件および手順を説明します。



重要

Red Hat は、プロビジョニングネットワークでのみ IPMI および PXE をサポートします。Red Hat は、IBM Cloud® デプロイメントで Secure Boot などの Red Hat Fish、仮想メディア、またはその他の補完テクノロジーをテストしていません。プロビジョニングネットワークが必要です。

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールには、以下が必要です。

- プロビジョナーを実行するための、Red Hat Enterprise Linux CoreOS (RHCOS) 8.x がインストールされた 1 つのノード
- 3 つのコントロールプレーンノード
- 1 つのルーティング可能なネットワーク
- 1 つのプロビジョニングネットワーク

IBM Cloud® Bare Metal (Classic) に、インストーラーでプロビジョニングされる OpenShift Container Platform のインストールを開始する前に、以下の前提条件および要件に対応します。

17.1.1. IBM Cloud Bare Metal (Classic) インフラストラクチャーのセットアップ

OpenShift Container Platform クラスターを IBM Cloud® Bare Metal (Classic) にデプロイするには、最初に IBM Cloud® ノードをプロビジョニングする必要があります。

重要

Red Hat は、**provisioning** ネットワークでのみ IPMI および PXE をサポートします。Red Hat は、IBM Cloud® デプロイメントで Secure Boot などの Red Hat Fish、仮想メディア、またはその他の補完テクノロジーをテストしていません。**provisioning** ネットワークが必要です。

IBM Cloud® API を使用して、IBM Cloud® ノードをカスタマイズできます。IBM Cloud® ノードを作成する場合は、以下の要件を考慮する必要があります。

クラスターごとにデータセンターを 1 つ使用します。
OpenShift Container Platform クラスターのすべてのノードは、同じ IBM Cloud® データセンターで実行する必要があります。

パブリックおよびプライベート VLAN を作成します。
1 つのパブリック VLAN と単一のプライベート VLAN を持つすべてのノードを作成します。

サブネットに十分な IP アドレスがあることを確認します。
IBM Cloud® パブリック VLAN サブネットは、デフォルトで **/28** 接頭辞を使用し、16 個の IP アドレスを

提供します。これは、3 つのコントロールプレーンノード、4 つのワーカーノード、および **baremetal** ネットワーク上 2 つの IP アドレス (API VIP および Ingress VIP) で設定されるクラスターには十分です。大規模なクラスターの場合は、接頭辞が小さい可能性があります。

IBM Cloud® private VLAN サブネットは、デフォルトで **/26** 接頭辞を使用し、64 個の IP アドレスを提供します。IIBM Cloud® Bare Metal (Classic) はプライベートネットワーク IP アドレスを使用して、各ノードの Baseboard Management Controller (BMC) にアクセスします。OpenShift Container Platform は、**provisioning** ネットワークの追加サブネットを作成します。プライベート VLAN を使用した **provisioning** ネットワークのサブネットのルートに対するネットワークトラフィック。大規模なクラスターの場合は、接頭辞が小さい可能性があります。

表17.1 接頭辞ごとの IP アドレス

IP アドレス	接頭辞
32	**/27**
64	**/26**
128	**/25**
256	**/24**

NIC の設定
OpenShift Container Platform は、2 つのネットワークを使用してデプロイします。

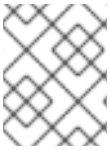
- **provisioning**: **provisioning** ネットワークは OpenShift Container Platform クラスターの一部である基礎となるオペレーティングシステムを各ノードにプロビジョニングするために使用されるルーティング不可能なネットワークです。
- **baremetal**: **baremetal** ネットワークはルーティング可能なネットワークです。任意の NIC 順序を使用して **baremetal** ネットワークと対話することができます。ただし、**provisioning** ネットワーク用の **provisioningNetworkInterface** 設定で指定される NIC ではなく、ノードの **bootMACAddress** 設定に関連する NIC であることが条件となります。

クラスターノードには 3 つ以上の NIC を追加できますが、インストールプロセスでは最初の 2 つの NIC のみに焦点が当てられます。以下に例を示します。

NIC	ネットワーク	VLAN
NIC1	**provisioning**	<provisioning_vlan>
NIC2	**baremetal**	<baremetal_vlan>

上記の例では、すべてのコントロールプレーンおよびワーカーノードの NIC1 は、OpenShift Container Platform クラスターのインストールにのみ使用されるルーティング不可能なネットワーク (**provisioning**) に接続します。すべてのコントロールプレーンおよびワーカーノードの NIC2 は、ルーティング可能な **baremetal** ネットワークに接続されます。

PXE	ブート順序
NIC1 PXE 対応の **provisioning** ネットワーク	1
NIC2 **baremetal** ネットワーク	2



注記

PXE が **provisioning** ネットワークに使用する NIC で有効になっており、他のすべての NIC で無効になっていることを確認します。

正規名の設定
クライアントは、**baremetal** ネットワークで OpenShift Container Platform クラスターにアクセスします。正規名の拡張がクラスター名である IBM Cloud® サブドメインまたはサブゾーンを設定します。

```
<cluster_name>.<domain>
```

以下に例を示します。

```
test-cluster.example.com
```

DNS エントリーの作成
以下について、パブリックサブネットで未使用の IP アドレスを解決する DNS **A** レコードエントリーを作成する必要があります。

使用法	ホスト名	IP
API	api.<cluster_name>.<domain>	<ip>
Ingress LB (アプリケーション)	*.apps.<cluster_name>.<domain>	<ip>

コントロールプレーンおよびワーカーノードは、プロビジョニング後にすでに DNS エントリーを持っています。

以下の表は、完全修飾ドメイン名の例を示しています。API および Nameserver アドレスは、正式名の拡張子で始まります。コントロールプレーンおよびワーカーノードのホスト名は例であるため、任意のホストの命名規則を使用することができます。

使用法	ホスト名	IP
API	api.<cluster_name>.<domain>	<ip>
Ingress LB (アプリケーション)	*.apps.<cluster_name>.<domain>	<ip>
プロビジョナーノード	provisioner.<cluster_name>.<domain>	<ip>

使用法	ホスト名	IP
Master-0	openshift-master-0.<cluster_name>.<domain>	<ip>
Master-1	openshift-master-1.<cluster_name>.<domain>	<ip>
Master-2	openshift-master-2.<cluster_name>.<domain>	<ip>
Worker-0	openshift-worker-0.<cluster_name>.<domain>	<ip>
Worker-1	openshift-worker-1.<cluster_name>.<domain>	<ip>
Worker-n	openshift-worker-n.<cluster_name>.<domain>	<ip>

OpenShift Container Platform には、クラスターメンバーシップ情報を使用して **A** レコードを生成する機能が含まれます。これにより、ノード名が IP アドレスに解決されます。ノードが API に登録されると、クラスターは CoreDNS-mDNS を使用せずにこれらのノード情報を分散できます。これにより、マルチキャスト DNS に関連付けられたネットワークトラフィックがなくなります。

重要

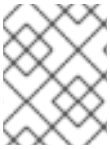
IBM Cloud® ノードのプロビジョニング後に、CoreDNS を削除するとローカルエントリーが非表示になるので、**api.<cluster_name>.<domain>** ドメイン名の DNS エントリーを作成する必要があります。外部 DNS サーバーで **api.<cluster_name>.<domain>** ドメイン名の DNS レコードの作成に失敗すると、ワーカーノードがクラスターに参加できなくなります。

ネットワークタイムプロトコル (NTP)
クラスター内の各 OpenShift Container Platform ノードは NTP サーバーにアクセスできる必要があります。OpenShift Container Platform ノードは NTP を使用してクロックを同期します。たとえば、クラスターノードは、検証を必要とする SSL 証明書を使用します。これは、ノード間の日付と時刻が同期していない場合に失敗する可能性があります。

重要

各クラスターノードの BIOS 設定で一貫性のあるクロックの日付と時刻の形式を定義してください。そうしないと、インストールが失敗する可能性があります。

DHCP サーバーの設定
IBM Cloud® Bare Metal (Classic) は、パブリックまたはプライベートの VLAN で DHCP を実行しません。IBM Cloud® ノードのプロビジョニング後に、OpenShift Container Platform の **baremetal** ネットワークに対応するパブリック VLAN の DHCP サーバーを設定する必要があります。



注記

各ノードに割り当てられる IP アドレスは、IBM Cloud® Bare Metal (Classic) プロビジョニングシステムによって割り当てられる IP アドレスに一致する必要はありません。

詳細は、Configuring the public subnet セクションを参照してください。

BMC アクセス権限の確認
ダッシュボードの各ノードの Remote management ページには、ノードのインテリジェントなプラットフォーム管理インターフェイス (IPMI) 認証情報が含まれます。デフォルトの IPMI 権限により、ユーザーが特定のブートターゲットの変更を阻止します。Ironic が変更を加えることができるように、特権レベルを **OPERATOR** に変更する必要があります。

install-config.yaml ファイルで、各 BMC の設定に使用される URL に **privilegelevel** パラメーターを追加します。詳細は、Configuring the install-config.yaml file セクションを参照してください。以下に例を示します。

```
ipmi://<IP>:<port>?privilegelevel=OPERATOR
```

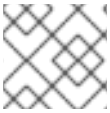
または、IBM Cloud® サポートに連絡し、各ノードの **ADMINISTRATOR** に対する IPMI 特権を増やすように要求します。

ベアメタルサーバーの作成
Create resource→ Bare Metal Servers for Classicに移動して、IBM Cloud® ダッシュボード にベアメタルサーバーを作成します。

または、**ibmcloud** CLI ユーティリティーを使用してベアメタルサーバーを作成できます。以下に例を示します。

```
$ ibmcloud sl hardware create --hostname <SERVERNAME> \
                              --domain <DOMAIN> \
                              --size <SIZE> \
                              --os <OS-TYPE> \
                              --datacenter <DC-NAME> \
                              --port-speed <SPEED> \
                              --billing <BILLING>
```

IBM Cloud® CLI のインストールに関する詳細は、Installing the stand-alone IBM Cloud® CLI を参照してください。



注記

IBM Cloud® Server が利用可能な状態になるまで 3-5 時間かかる場合があります。

17.2. OPENSHIFT CONTAINER PLATFORM インストールの環境の設定

17.2.1. IBM Cloud(R) Bare Metal (Classic) インフラストラクチャー上でプロビジョナーノードを準備する

provisioner ノードを準備するには、以下の手順を実行します。

手順

1. **ssh** でプロビジョナーノードにログインします。
2. root 以外のユーザー (**kni**) を作成し、そのユーザーに **sudo** 権限を付与します。

```
# useradd kni
```

```
# passwd kni
```

```
# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
```

```
# chmod 0440 /etc/sudoers.d/kni
```

3. 新規ユーザーの **ssh** キーを作成します。

```
# su - kni -c "ssh-keygen -f /home/kni/.ssh/id_rsa -N ""
```

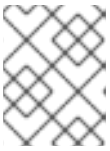
4. プロビジョナーノードで新規ユーザーとしてログインします。

```
# su - kni
```

5. Red Hat Subscription Manager を使用してプロビジョナーノードを登録します。

```
$ sudo subscription-manager register --username=<user> --password=<pass> --auto-attach
```

```
$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-appstream-rpms \
                                  --enable=rhel-8-for-x86_64-baseos-rpms
```



注記

Red Hat Subscription Manager についての詳細は、Using and Configuring Red Hat Subscription Manager を参照してください。

6. 以下のパッケージをインストールします。

```
$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```

7. ユーザーを変更して、新たに作成したユーザーに **libvirt** グループを追加します。

```
$ sudo usermod --append --groups libvirt kni
```

8. **firewalld** を起動します。

```
$ sudo systemctl start firewalld
```

9. **firewalld** を有効にします。

```
$ sudo systemctl enable firewalld
```

10. **http** サービスを起動します。

```
$ sudo firewall-cmd --zone=public --add-service=http --permanent
```

```
$ sudo firewall-cmd --reload
```

11. **libvirtd** サービスを開始して、これを有効にします。

```
$ sudo systemctl enable libvirtd --now
```

12. プロビジョナーノードの ID を設定します。

```
$ PRVN_HOST_ID=<ID>
```

以下の **ibmcloud** コマンドで ID を表示できます。

```
$ ibmcloud sl hardware list
```

13. パブリックサブネットの ID を設定します。

```
$ PUBLICSUBNETID=<ID>
```

以下の **ibmcloud** コマンドで ID を表示できます。

```
$ ibmcloud sl subnet list
```

14. プライベートサブネットの ID を設定します。

```
$ PRIVSUBNETID=<ID>
```

以下の **ibmcloud** コマンドで ID を表示できます。

```
$ ibmcloud sl subnet list
```

15. provisioner ノードのパブリック IP アドレスを設定します。

```
$ PRVN_PUB_IP=$(ibmcloud sl hardware detail $PRVN_HOST_ID --output JSON | jq
.primaryIpAddress -r)
```

16. パブリックネットワークの CIDR を設定します。

```
$ PUBLICCIDR=$(ibmcloud sl subnet detail $PUBLICSUBNETID --output JSON | jq .cidr)
```

17. パブリックネットワークの IP アドレスおよび CIDR を設定します。

```
$ PUB_IP_CIDR=$PRVN_PUB_IP/$PUBLICCIDR
```

18. パブリックネットワークのゲートウェイを設定します。

```
$ PUB_GATEWAY=$(ibmcloud sl subnet detail $PUBLICSUBNETID --output JSON | jq
.gateway -r)
```

19. プロビジョナーノードのプライベート IP アドレスを設定します。

```
$ PRVN_PRIV_IP=$(ibmcloud sl hardware detail $PRVN_HOST_ID --output JSON | \
        jq .primaryBackendIpAddress -r)
```

20. プライベートネットワークの CIDR を設定します。

```
$ PRIVCIDR=$(ibmcloud sl subnet detail $PRIVSUBNETID --output JSON | jq .cidr)
```

21. プライベートネットワークの IP アドレスおよび CIDR を設定します。

```
$ PRIV_IP_CIDR=$PRVN_PRIV_IP/$PRIVCIDR
```

22. プライベートネットワークのゲートウェイを設定します。

```
$ PRIV_GATEWAY=$(ibmcloud sl subnet detail $PRIVSUBNETID --output JSON | jq
.gateway -r)
```

23. **baremetal** および **provisioning** ネットワークのブリッジを設定します。

```
$ sudo nohup bash -c "
    nmcli --get-values UUID con show | xargs -n 1 nmcli con delete
    nmcli connection add ifname provisioning type bridge con-name provisioning
    nmcli con add type bridge-slave ifname eth1 master provisioning
    nmcli connection add ifname baremetal type bridge con-name baremetal
    nmcli con add type bridge-slave ifname eth2 master baremetal
    nmcli connection modify baremetal ipv4.addresses $PUB_IP_CIDR ipv4.method manual
ipv4.gateway $PUB_GATEWAY
    nmcli connection modify provisioning ipv4.addresses 172.22.0.1/24,$PRIV_IP_CIDR
ipv4.method manual
    nmcli connection modify provisioning +ipv4.routes \"10.0.0.0/8 $PRIV_GATEWAY\"
    nmcli con down baremetal
    nmcli con up baremetal
    nmcli con down provisioning
    nmcli con up provisioning
    init 6
"
```

注記

eth1 および **eth2** の場合は、必要に応じて適切なインターフェイス名を置き換えます。

24. 必要な場合は、**provisioner** ノードに対して再度 SSH を実行します。

```
# ssh kni@provisioner.<cluster-name>.<domain>
```

25. 接続ブリッジが適切に作成されていることを確認します。

```
$ sudo nmcli con show
```

出力例

```
NAME              UUID                                   TYPE      DEVICE
baremetal         4d5133a5-8351-4bb9-bfd4-3af264801530   bridge    baremetal
provisioning      43942805-017f-4d7d-a2c2-7cb3324482ed   bridge    provisioning
virbr0            d9bca40f-eee1-410b-8879-a2d4bb0465e7   bridge    virbr0
bridge-slave-eth1 76a8ed50-c7e5-4999-b4f6-6d9014dd0812   ethernet  eth1
bridge-slave-eth2 f31c3353-54b7-48de-893a-02d2b34c4736   ethernet  eth2
```

26. **pull-secret.txt** ファイルを作成します。

```
$ vim pull-secret.txt
```

Web ブラウザーで、Install on Bare Metal with user-provisioned infrastructure に移動します。ステップ 1 で、**Download pull secret** をクリックします。**pull-secret.txt** ファイルにコンテンツを貼り付け、そのコンテンツを **kni** ユーザーのホームディレクトリーに保存します。

17.2.2. パブリックサブネットの設定

すべての OpenShift Container Platform クラスターノードはパブリックサブネット上になければなりません。IBM Cloud® Bare Metal (Classic) は、サブネット上に DHCP サーバーを提供しません。プロビジョナーノードで個別に設定します。

プロビジョナーノードの準備時に定義された BASH 変数をリセットする必要があります。準備後にプロビジョナーノードを再起動すると、BASH 変数が以前に設定された変数が削除されます。

手順

1. **dnsmasq** をインストールします。

```
$ sudo dnf install dnsmasq
```

2. **dnsmasq** 設定ファイルを開きます。

```
$ sudo vi /etc/dnsmasq.conf
```

3. 以下の設定を **dnsmasq** 設定ファイルに追加します。

```
interface=baremetal
except-interface=lo
bind-dynamic
log-dhcp

dhcp-range=<ip_addr>,<ip_addr>,<pub_cidr> 1
dhcp-option=baremetal,121,0.0.0.0/0,<pub_gateway>,<prvn_priv_ip>,<prvn_pub_ip> 2

dhcp-hostsfile=/var/lib/dnsmasq/dnsmasq.hostsfile
```

1. DHCP 範囲を設定します。**<ip_addr>** の両方のインスタンスをパブリックサブネットから 1 つの未使用の IP アドレスに置き換え、**baremetal** ネットワークの **dhcp-range** が開始し、IP アドレスで終了するようにします。**<pub_cidr>** をパブリックサブネットの CIDR に置き換えます。

2.

DHCP オプションを設定します。**<pub_gateway>** を、**baremetal** ネットワークのゲートウェイの IP アドレスに置き換えます。**<prvn_priv_ip>** を **provisioning** ネットワーク上

<pub_cidr> の値を取得するには、以下を実行します。

```
$ ibmcloud sl subnet detail <publicsubnetid> --output JSON | jq .cidr
```

<publicsubnetid> をパブリックサブネットの ID に置き換えます。

<pub_gateway> の値を取得するには、以下を実行します。

```
$ ibmcloud sl subnet detail <publicsubnetid> --output JSON | jq .gateway -r
```

<publicsubnetid> をパブリックサブネットの ID に置き換えます。

<prvn_priv_ip> の値を取得するには、以下を実行します。

```
$ ibmcloud  sl hardware detail <id> --output JSON | \
          jq .primaryBackendIpAddress -r
```

<id> をプロビジョナーノードの ID に置き換えます。

<prvn_pub_ip> の値を取得するには、以下を実行します。

```
$ ibmcloud sl hardware detail <id> --output JSON | jq .primaryIpAddress -r
```

<id> をプロビジョナーノードの ID に置き換えます。

4. クラスターのハードウェアのリストを取得します。

```
$ ibmcloud sl hardware list
```

5. 各ノードの MAC アドレスおよび IP アドレスを取得します。

```
$ ibmcloud sl hardware detail <id> --output JSON | \
  jq '.networkComponents[] | \
  "\(.primaryIpAddress) \(.macAddress)"' | grep -v null
```

<id> をノードの ID に置き換えてください。

出力例

```
"10.196.130.144 00:e0:ed:6a:ca:b4"
"141.125.65.215 00:e0:ed:6a:ca:b5"
```

パブリックネットワークの MAC アドレスと IP アドレスを書き留めておきます。プライベートネットワークの MAC アドレスを別々に書留ておきます。これは、後に **install-config.yaml** ファイルで使用します。各ノードにパブリック **baremetal** ネットワークのパブリック MAC アドレスと IP アドレスがすべてあり、プライベートの **provisioning** の MAC アドレスになるまで、この手順を繰り返します。

6. 各ノードのパブリック **baremetal** ネットワークの MAC アドレスと IP アドレスペアを **dnsmasq.hostsfile** ファイルに追加します。

```
$ sudo vim /var/lib/dnsmasq/dnsmasq.hostsfile
```

入力の例

```
00:e0:ed:6a:ca:b5,141.125.65.215,master-0
<mac>,<ip>,master-1
<mac>,<ip>,master-2
<mac>,<ip>,worker-0
<mac>,<ip>,worker-1
...
```

<mac>,<ip> を、対応するノード名のパブリック MAC アドレスとパブリック IP アドレスに置き換えます。

7. **dnsmasq** を開始します。

```
$ sudo systemctl start dnsmasq
```

8. **dnsmasq** を有効にして、ノードのブート時に起動できるようにします。

```
$ sudo systemctl enable dnsmasq
```

9. **dnsmasq** が実行中であることを確認します。

```
$ sudo systemctl status dnsmasq
```

出力例

```
● dnsmasq.service - DNS caching server.
Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; enabled; vendor preset: disabled)
Active: active (running) since Tue 2021-10-05 05:04:14 CDT; 49s ago
Main PID: 3101 (dnsmasq)
Tasks: 1 (limit: 204038)
Memory: 732.0K
CGroup: /system.slice/dnsmasq.service
└─3101 /usr/sbin/dnsmasq -k
```

10. UDP プロトコルでポート **53** および **67** を開きます。

```
$ sudo firewall-cmd --add-port 53/udp --permanent
```

```
$ sudo firewall-cmd --add-port 67/udp --permanent
```

11. masquerade を使用して、外部ゾーンに **provisioning** を追加します。

```
$ sudo firewall-cmd --change-zone=provisioning --zone=external --permanent
```

このステップにより、管理サブネットへの IPMI 呼び出しのネットワークアドレス変換が確保されます。

12. **firewalld** 設定を再読み込みします。

```
$ sudo firewall-cmd --reload
```

17.2.3. OpenShift Container Platform インストーラーの取得

インストールプログラムの **stable-4.x** バージョンと選択したアーキテクチャーを使用して、OpenShift Container Platform の一般公開の安定バージョンをデプロイします。

```
$ export VERSION=stable-4.16
```

```
$ export RELEASE_ARCH=<architecture>
```

```
$ export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-v4/$RELEASE_ARCH/clients/ocp/$VERSION/release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
```

17.2.4. OpenShift Container Platform インストールのデプロイメント

インストーラーを取得したら、インストーラーを展開します。

手順

1. 環境変数を設定します。

```
$ export cmd=openshift-baremetal-install
```

```
$ export pullsecret_file=~/pull-secret.txt
```

```
$ export extract_dir=$(pwd)
```

2. **oc** バイナリーを取得します。

```
$ curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/openshift-client-linux.tar.gz | tar zxvf - oc
```

3. インストーラーを実行します。

```
$ sudo cp oc /usr/local/bin
```

```
$ oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to "${extract_dir}" ${RELEASE_IMAGE}
```

```
$ sudo cp openshift-baremetal-install /usr/local/bin
```

17.2.5. install-config.yaml ファイルの設定

install-config.yaml ファイルには、追加の詳細情報が必要です。それらの情報のほとんどは、インストーラーおよび結果として作成されるクラスターが利用可能な IBM Cloud® Bare Metal (Classic) ハードウェアを完全に管理するのに必要な利用可能なハードウェアについての十分な情報として提供されま

す。ベアメタルへのインストールと IBM Cloud Bare Metal (Classic) へのインストールの相違点は **install-config.yaml** ファイルの BMC セクションで IPMI の特権レベルを明示的に設定する必要があることです。

手順

1. **install-config.yaml** を設定します。**pullSecret** および **sshKey** など、環境に合わせて適切な変数を変更します。

```
apiVersion: v1
baseDomain: <domain>
metadata:
  name: <cluster_name>
networking:
  machineNetwork:
  - cidr: <public-cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIP: <api_ip>
    ingressVIP: <wildcard_ip>
    provisioningNetworkInterface: <NIC1>
    provisioningNetworkCIDR: <CIDR>
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://10.196.130.145?privilegelevel=OPERATOR (1)
          username: root
          password: <password>
        bootMACAddress: 00:e0:ed:6a:ca:b4 (2)
        rootDeviceHints:
          deviceName: "/dev/sda"
      - name: openshift-worker-0
        role: worker
        bmc:
          address: ipmi://<out-of-band-ip>?privilegelevel=OPERATOR (3)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1_mac_address> (4)
        rootDeviceHints:
          deviceName: "/dev/sda"
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

(1) (3) **bmc.address** は、値が **OPERATOR** に設定された **privilegelevel** の設定を提供します。これは、IBM Cloud® Bare Metal (Classic) インフラストラクチャーに必要です。

24 対応するノードのプライベート **provisioning** ネットワーク NIC の MAC アドレスを追加します。

注記

ibmcloud コマンドラインユーティリティーを使用して、パスワードを取得できます。

```
$ ibmcloud sl hardware detail <id> --output JSON | \
  jq '"(.networkManagementIpAddress)
(.remoteManagementAccounts[0].password)"'
```

<id> をノードの ID に置き換えてください。

2. クラスター設定を保存するディレクトリーを作成します。

```
$ mkdir ~/clusterconfigs
```

3. **install-config.yaml** ファイルをディレクトリーにコピーします。

```
$ cp install-config.yaml ~/clusterconfig
```

4. OpenShift Container Platform クラスターをインストールする前に、すべてのベアメタルノードの電源がオフになっていることを確認します。

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management_server_ip> power off
```

5. 以前に試行したデプロイメントにより古いブートストラップリソースが残っている場合は、これを削除します。

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

17.2.6. 追加の install-config パラメーター

install-config.yaml ファイルに必要なパラメーター **hosts** パラメーターおよび **bmc** パラメーターについては、以下の表を参照してください。

表17.2 必須パラメーター

パラメーター	デフォルト	説明
baseDomain		クラスターのドメイン名。例: **example.com**

パラメーター	デフォルト	説明
bootMode	**UEFI**	ノードのブートモード。オプションは、**legacy**、**UEFI**、および **UEFISecureBoot** です。**bootMode** が設定されていない場合は、ノードを検査する間に Ironic がこれを設定します。
bootstrapExternalStaticDNS		ブートストラップノードの静的ネットワーク DNS。ベアメタルネットワークに Dynamic Host Configuration Protocol (DHCP) サーバーがない場合に、静的 IP アドレスを使用してクラスターをデプロイする場合は、この値を設定する必要があります。この値を設定しないと、インストールプログラムが **bootstrapExternalStaticGateway** の値を使用します。そのため、ゲートウェイと DNS の IP アドレス値が異なる場合に問題が発生します。
bootstrapExternalStaticIP		ブートストラップ VM の静的 IP アドレス。ベアメタルネットワークに DHCP サーバーがない場合に、静的 IP アドレスを使用してクラスターをデプロイする場合は、この値を設定する必要があります。
bootstrapExternalStaticGateway		ブートストラップ VM のゲートウェイの静的 IP アドレス。ベアメタルネットワークに DHCP サーバーがない場合に、静的 IP アドレスを使用してクラスターをデプロイする場合は、この値を設定する必要があります。
sshKey		**sshKey** 設定には、コントロールプレーンノードとコンピュートノードにアクセスするために必要な、**~/.ssh/id_rsa.pub** ファイル内のキーを含めます。通常、このキーは **provisioner** ノードのキーです。
pullSecret		**pullSecret** 設定には、プロビジョナーノードの準備の際に Install OpenShift on Bare Metal ページからダウンロードしたプルシークレットのコピーが含まれます。
metadata: name:		OpenShift Container Platform クラスターに指定される名前。例: **openshift**
networking: machineNetwork: - cidr:		外部ネットワークの公開 CIDR (Classless Inter-Domain Routing)。例: **10.0.0.0/24** など。
compute: - name: worker		OpenShift Container Platform クラスターでは、ノードがゼロの場合でも、コンピュートノードに名前を付ける必要があります。

パラメーター	デフォルト	説明
compute: replicas: 2		replicas は、OpenShift Container Platform クラスターのコンピュートノードの数を設定します。
controlPlane: name: master		OpenShift Container Platform クラスターでは、コントロールプレーンノードに名前が必要です。
controlPlane: replicas: 3		replicas は、OpenShift Container Platform クラスターの一部として含まれるコントロールプレーンノードの数を設定します。
provisioningNetworkInterface		ベアメタルネットワークに接続されたノード上のネットワークインターフェイス名。OpenShift Container Platform 4.9 以降のリリースのために、NIC の名前を識別するために **provisioningNetworkInterface** 設定を使用する代わりに、Ironic が NIC の IP アドレスを識別できるように **bootMACAddress** 設定を使用します。
defaultMachinePlatform		プラットフォーム設定なしでマシンプールに使用されるデフォルト設定。
apiVIPs		(オプション) Kubernetes API 通信の仮想 IP アドレス。 この設定は、Machine Network からの予約済み IP として **install-config.yaml** ファイルで提供するか、デフォルト名が正しく解決されるように DNS で事前設定する必要があります。**install-config.yaml** ファイルの **apiVIPs** 設定に値を追加するときは、FQDN ではなく仮想 IP アドレスを使用します。デュアルスタックネットワークを使用する場合には、プライマリー IP アドレスは IPv4 ネットワークからのものである必要があります。設定されていない場合、インストールプログラムは **api.<cluster_name>.<base_domain>** を使用して DNS から IP アドレスを取得します。  **注記** OpenShift Container Platform 4.12 より前では、クラスターのインストールプログラムは **apiVIP** 設定の IPv4 アドレスまたは IPv6 アドレスのみを受け入れていました。OpenShift Container Platform 4.12 以降から、**apiVIP** 設定は非推奨になりました。代わりに、**apiVIPs** 設定のリスト形式を使用して、IPv4 アドレス、IPv6 アドレス、または両方の IP アドレス形式を指定します。

パラメーター	デフォルト	説明
disableCertificateVerification	**False**	**redfish** および **redfish-virtualmedia** では、このパラメーターで BMC アドレスを管理する必要があります。BMC アドレスに自己署名証明書を使用する場合は、値が **True** である必要があります。
ingressVIPs		(オプション) Ingress トラフィックの仮想 IP アドレス。 この設定は、Machine Network からの予約済み IP として **install-config.yaml** ファイルで提供するか、デフォルト名が正しく解決されるように DNS で事前設定する必要があります。**install-config.yaml** ファイルの **ingressVIPs** 構成設定に値を追加するときは、FQDN ではなく仮想 IP アドレスを使用してください。デュアルスタックネットワークを使用する場合には、プライマリー IP アドレスは IPv4 ネットワークからのものである必要があります。設定されていない場合、インストールプログラムは **test.apps.<cluster_name>.<base_domain>** を使用して DNS から IP アドレスを取得します。  **注記** OpenShift Container Platform 4.12 より前では、クラスターのインストールプログラムは **ingressVIP** 設定の IPv4 アドレスまたは IPv6 アドレスのみを受け入れていました。OpenShift Container Platform 4.12 以降では、**ingressVIP** 設定は非推奨です。代わりに、**ingressVIPs** 設定のリスト形式を使用して、IPv4 アドレス、IPv6 アドレス、または両方の IP アドレス形式を指定します。

表17.3 オプションのパラメーター

パラメーター	デフォルト	説明
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	プロビジョニングネットワークでノードの IP 範囲を定義します。
provisioningNetworkCIDR	**172.22.0.0/24**	プロビジョニングに使用するネットワークの CIDR。このオプションは、プロビジョニングネットワークでデフォルトのアドレス範囲を使用しない場合に必要です。
clusterProvisioningIP	**provisioningNetworkCIDR** の 3 番目の IP アドレス。	プロビジョニングサービスが実行されるクラスター内の IP アドレス。デフォルトは、プロビジョニングサブネットの 3 番目の IP アドレスに設定されます。例: **172.22.0.3**.

パラメーター	デフォルト	説明
bootstrapProvisioningIP	**provisioningNetworkCIDR** の 2 番目の IP アドレス	インストーラーがコントロールプレーン (マスター) ノードをデプロイしている間にプロビジョニングサービスが実行されるブートストラップ仮想マシンの IP アドレス。デフォルトは、プロビジョニングサブネットの 2 番目の IP アドレスに設定されます。例: **172.22.0.2** または **2620:52:0:1307::2**。
externalBridge	**baremetal**	ベアメタルネットワークに接続されたハイパーバイザーのベアメタルブリッジの名前。
provisioningBridge	**provisioning**	プロビジョニングネットワークに接続されている **provisioner** ホストのプロビジョニングブリッジの名前。
architecture		クラスターのホストアーキテクチャーを定義します。有効な値は **amd64** または **arm64** です。
defaultMachinePlatform		プラットフォーム設定なしでマシンプールに使用されるデフォルト設定。
bootstrapOSImage		ブートストラップノードのデフォルトのオペレーティングシステムイメージを上書きするための URL。URL にはイメージの SHA-256 ハッシュが含まれている必要があります。例: **https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>**。
provisioningNetwork		**provisioningNetwork** 設定は、クラスターがプロビジョニングネットワークを使用するかどうかを決定します。存在する場合、設定はクラスターがネットワークを管理するかどうかも決定します。 **Disabled**: プロビジョニングネットワークの要件を無効にするには、このパラメーターを **Disabled** に設定します。**Disabled** に設定すると、仮想メディアベースのプロビジョニングのみを使用するか、アシステッドインストーラーを使用してクラスターを起動する必要があります。**Disabled**にして、電源管理を使用する場合、BMC はベアメタルネットワークからアクセスできる必要があります。**Disabled** の場合は、プロビジョニングサービスに使用されるベアメタルネットワークで 2 つの IP アドレスを指定する必要があります。 **Managed**: DHCP、TFTP などを含むプロビジョニングネットワークを完全に管理するには、このパラメーターを **Managed**(デフォルト) に設定します。 **Unmanaged**: このパラメーターを **Unmanaged** に設定してプロビジョニングネットワークを引き続き有効にしますが、DHCP の手動設定を行います。仮想メディアのプロビジョニングが推奨されますが、必要に応じて PXE を引き続き利用できます。

パラメーター	デフォルト	説明
httpProxy		このパラメーターを、環境内で使用する適切な HTTP プロキシーに設定します。
httpsProxy		このパラメーターを、環境内で使用する適切な HTTPS プロキシーに設定します。
noProxy		このパラメーターを、環境内のプロキシーの使用に対する例外のリストに設定します。

ホスト
hosts パラメーターは、クラスターのビルドに使用される個別のベアメタルアセットのリストです。

表17.4 ホスト

名前	デフォルト	説明
name		詳細情報に関連付ける **BareMetalHost** リソースの名前。例: **openshift-master-0**
role		ベアメタルノードのロール。**master** (コントロールプレーンノード) または **worker** (コンピュートノード) のいずれか。
bmc		ベースボード管理コントローラーの接続詳細。詳細は、BMC アドレス指定のセクションを参照してください。
bootMACAddress		ホストがプロビジョニングネットワークに使用する NIC の MAC アドレス。Ironic は、**bootMACAddress** 設定を使用して IP アドレスを取得します。次に、ホストにバインドします。 **注記** プロビジョニングネットワークを無効にした場合は、ホストから有効な MAC アドレスを提供する必要があります。
networkConfig		このオプションのパラメーターを設定して、ホストのネットワークインターフェイスを設定します。詳細については、オプション: ホストネットワークインターフェイスの設定を参照してください。

17.2.7. ルートデバイスのヒント

rootDeviceHints パラメーターは、インストーラーが Red Hat Enterprise Linux CoreOS (RHCOS) イメージを特定のデバイスにプロビジョニングできるようにします。インストーラーは、検出順にデバイ

スを検査し、検出された値をヒントの値と比較します。インストーラーは、ヒント値に一致する最初に検出されたデバイスを使用します。この設定は複数のヒントを組み合わせることができますが、デバイスは、インストーラーがこれを選択できるようにすべてのヒントに一致する必要があります。

表17.5 サブフィールド

サブフィールド	説明
deviceName	**/dev/vda** や **/dev/disk/by-path/** などの Linux デバイス名を含む文字列。ストレージの場所への **/dev/disk/by-path/<device_path>** リンクを使用することを推奨します。ヒントは、実際の値と完全に一致する必要があります。
hctl	**0:0:0:0** などの SCSI バスアドレスを含む文字列。ヒントは、実際の値と完全に一致する必要があります。
model	ベンダー固有のデバイス識別子を含む文字列。ヒントは、実際の値のサブ文字列になります。
vendor	デバイスのベンダーまたは製造元の名前が含まれる文字列。ヒントは、実際の値のサブ文字列になります。
serialNumber	デバイスのシリアル番号を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
minSizeGigabytes	デバイスの最小サイズ (ギガバイト単位) を表す整数。
wwn	一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
wwnWithExtension	ベンダー拡張が追加された一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
wwnVendorExtension	一意のベンダーストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
rotational	デバイスがローテーションするディスクである (true) か、そうでないか (false) を示すブール値。

使用例

```
- name: master-0
  role: master
  bmc:
    address: ipmi://10.10.0.3:6203
    username: admin
```

```
  password: redhat
bootMACAddress: de:ad:be:ef:00:40
rootDeviceHints:
  deviceName: "/dev/sda"
```

17.2.8. OpenShift Container Platform マニフェストの作成

1. OpenShift Container Platform マニフェストを作成します。

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs create manifests
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
WARNING Discarding the OpenShift Manifest that was provided in the target directory
because its dependencies are dirty and it needs to be regenerated
```

17.2.9. OpenShift Container Platform インストーラーを使用したクラスターのデプロイ

OpenShift Container Platform インストーラーを実行します。

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create cluster
```

17.2.10. インストール後

デプロイメントプロセスで、**tail** コマンドを install ディレクトリーフォルダーの **.openshift_install.log** ログファイルに対して実行して、インストールの全体のステータスを確認できます。

```
$ tail -f /path/to/install-dir/.openshift_install.log
```

第18章 IBM Z および IBM LINUXONE へのインストール

18.1. IBM Z および IBM LINUXONE へのインストールの準備

18.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

このドキュメントは IBM Z® のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。

18.1.2. OpenShift Container Platform の IBM Z または IBM& LinuxONE へのインストール方法の選択

OpenShift Container Platform インストールプログラムでは、次の方法で IBM Z® にクラスターをデプロイできます。

- **インタラクティブ**: Web ベースの Assisted Installer を使用してクラスターをデプロイできます。この方法はインストーラーのセットアップを必要とせず、IBM Z® のような接続された環境に最適です。
- **ローカルエージェントベース**: Agent-based Installer を使用してクラスターをローカルにデプロイできます。このインストーラーは、Assisted Installer の多くの利点を備えていますが、先にダウンロードして設定する必要があります。設定はコマンドラインインターフェイス (CLI) で行います。このアプローチは、非接続ネットワークに最適です。
- **完全な制御**: お客様が準備および保守するインフラストラクチャー にクラスターをデプロイできるため、最大限のカスタマイズ性が得られます。接続環境または非接続環境でクラスターをデプロイできます。

表18.1 IBM Z(R) のインストール方法

	Assisted Installer	Agent-based Installer	user-provisioned installation	installer-provisioned installation
z/VM を使用した IBM Z®	✓	✓	✓	
ネットワークが制限された環境の z/VM を使用した IBM Z®		✓	✓	
RHEL KVM を使用した IBM Z®	✓	✓	✓	
ネットワークが制限された環境の RHEL KVM を使用した IBM Z®		✓	✓	
IBM Z® の LPAR 内			✓	
ネットワークが制限された環境の IBM Z® の LPAR 内			✓	

インストールプロセスの詳細は、インストールプロセス を参照してください。

18.1.2.1. ユーザーによってプロビジョニングされるインフラストラクチャーでの BM Z への OpenShift Container Platform のインストール

ユーザーによってプロビジョニングされるインフラストラクチャーでは、ユーザーは OpenShift Container Platform に必要なすべてのリソースをプロビジョニングする必要があります。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自に提供するインフラストラクチャーでクラスターをインストールするには、IBM Z® プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

- **z/VM を使用したクラスターの IBM Z® および IBM® LinuxONE へのインストール**: 独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャーに、z/VM を使用した OpenShift Container Platform をインストールできます。
- **ネットワークが制限された環境での z/VM を使用したクラスターの IBM Z® および IBM® LinuxONE へのインストール**: インストールリリースコンテンツの内部ミラーを使用して、ネットワークが制限された環境または非接続環境で、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャーに z/VM を使用した OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

- **RHEL KVM を使用したクラスターの IBM Z® および IBM® LinuxONE へのインストール**: 独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャーに、KVM を使用した OpenShift Container Platform をインストールできます。

- **ネットワークが制限された環境での RHEL KVM を使用したクラスターの IBM Z® および IBM® LinuxONE へのインストール**: インストールリリースコンテンツの内部ミラーを使用して、ネットワークが制限された環境または非接続環境で、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャーに RHEL KVM を使用して OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

- **IBM Z® および IBM® LinuxONE 上の LPAR へのクラスターのインストール**: 独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャー上の論理パーティション (LPAR) に OpenShift Container Platform をインストールできます。

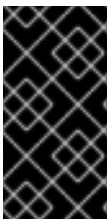
- **ネットワークが制限された環境での IBM Z® および IBM® LinuxONE 上の LPAR へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、ネットワークが制限された環境または非接続環境で、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャー上の LPAR に OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

18.2. Z/VM を使用したクラスターの IBM Z および IBM IBM® LINUXONE へのインストール

OpenShift Container Platform バージョン 4.16 では、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャーに、クラスターをインストールできます。

注記

このドキュメントは IBM Z® のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

18.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

18.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

18.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

18.2.3.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表18.2 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

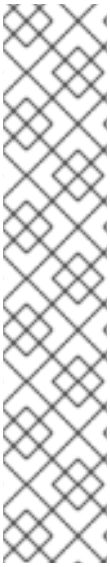
18.2.3.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表18.3 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
Compute	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

18.2.3.3. 最小の IBM Z システム環境

OpenShift Container Platform バージョン 4.16 は、以下の IBM® ハードウェアにインストールできます。

- IBM® z16 (すべてのモデル)、IBM® z15 (すべてのモデル)、IBM® z14 (すべてのモデル)
- IBM® LinuxONE 4 (すべてのモデル)、IBM® LinuxONE III (すべてのモデル)、IBM® LinuxONE Empire II、IBM® LinuxONE Rockhopper II

ハードウェア要件

- クラスターごとに、SMT2 対応の 6 つの Integrated Facilities for Linux (IFL) に相当します。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z® の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。

重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR のウェイト管理、エンタイトルメント、および CPU 共有が重要な役割を果たします。

オペレーティングシステム要件

- z/VM 7.2 以降の 1 インスタンス

z/VM インスタンスで、以下をセットアップします。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Z ネットワーク接続の要件
IBM Z® の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM vSwitch のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

ディスクストレージ

- FICON 接続のディスクストレージ (DASD)。これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

18.2.3.4. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。
- HiperSockets。ノードに直接割り当てたものか、z/VM ゲストに対して透過的になるように z/VM VSWITCH でブリッジしてノードに割り当てたもの。HiperSockets をノードに直接接続するには、RHEL 8 ゲストを介して外部ネットワークへのゲートウェイをセットアップし、HiperSockets ネットワークにブリッジする必要があります。

オペレーティングシステム要件

- 高可用性を確保する場合は z/VM 7.2 以降の 2 または 3 インスタンス

z/VM インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用に 3 ゲスト仮想マシン (z/VM インスタンスごとに 1 つ)
- OpenShift Container Platform コンピュートマシン用に 6 以上のゲスト仮想マシン (z/VM インスタンス全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン
- オーバーコミット環境で必須コンポーネントの可用性を確保するには、CP コマンドの **SET SHARE** を使用してコントロールプレーンの優先度を引き上げます。インフラストラクチャーノードが存在する場合は、同じ操作を行います。IBM® ドキュメントの SET SHARE を参照してください。

IBM Z ネットワーク接続の要件
IBM Z® の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM vSwitch のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

ディスクストレージ

- FICON 接続のディスクストレージ (DASD)。これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

18.2.3.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- IBM® ドキュメントの Bridging a HiperSockets LAN with a z/VM Virtual Switch を参照してください。

- パフォーマンスの最適化については、Scaling HyperPAV alias devices on Linux guests on z/VM を参照してください。
- LPAR のウェイト管理とエンタイトルメントについては、Topics in LPAR Performance を参照してください。
- IBM Z® および IBM® LinuxONE 環境に推奨されるホストプラクティス

18.2.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

18.2.3.6.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表18.4 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート

プロトコル	ポート	説明
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表18.5 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表18.6 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

18.2.3.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

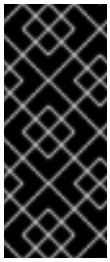
- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

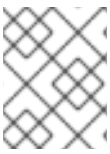
DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表18.7 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>. <base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>. <base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

18.2.3.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例18.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例18.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

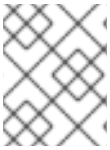
1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

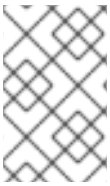


注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

18.2.3.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

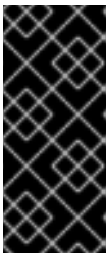


注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.8 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

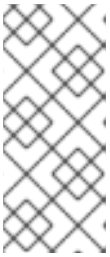
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.9 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

18.2.3.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例18.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
```

```
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (5)
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (6)
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

(1) ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

(2)(4) ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

(3) ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

(5) ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

(6) ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

18.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

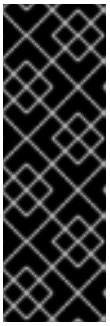
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。



重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

5. クラスターに必要な DNS インフラストラクチャーを設定します。

    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

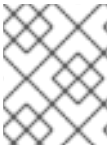
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

6. DNS 設定を検証します。

    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

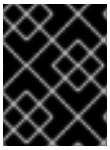


注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

18.2.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

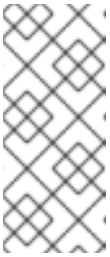
   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   注記

   出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

   random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

18.2.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、**~/.ssh/id_rsa** および **~/.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

18.2.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

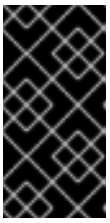
- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

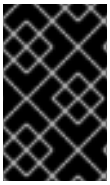
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

18.2.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

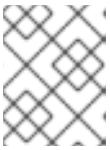
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

18.2.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Z® のインストール設定パラメーター

18.2.9.1. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture: s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture: s390x
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
```

```
  networkType: OVNKubernetes 11
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths": ...}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

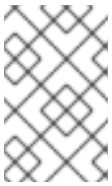
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が OpenShift Container Platform ノードで利用できない場合、**hyperthreading** パラメーターは影響を受けません。

重要

OpenShift Container Platform ノードまたは **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



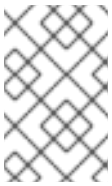
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネット



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Z® インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

18.2.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

18.2.9.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定される最小の 3 つのノードクラスターにゼロコンピューティングマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用

するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

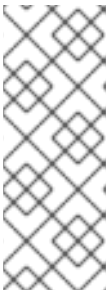
手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。



注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

18.2.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

18.2.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表18.10 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23

フィールド	型	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表18.11 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表18.12 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表18.13 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表18.14 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表18.15 **policyAuditConfig** オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表18.16 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

フィールド	型	説明
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表18.17 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表18.18 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表18.19 **ipsecConfig** オブジェクト

フィールド	型	説明

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表18.20 **kubeProxyConfig** オブジェクト

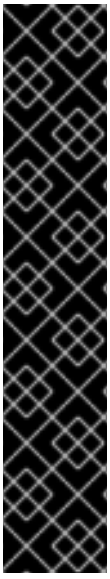
フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	型	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

18.2.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
```

```
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.2.12. IBM Z または IBM& LinuxONE 環境での静的 IP を使用した NBDE の設定

IBM Z® または IBM® LinuxONE 環境で NBDE ディスク暗号化を有効にするには、追加の手順が必要です。このセクションで詳しく説明します。

前提条件

- External Tang Server がセットアップされました。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。
- **butane** ユーティリティーをインストールしている。
- Butane でマシン設定を作成する手順を確認している。

手順

1. コントロールプレーンとコンピュートノードの Butane 設定ファイルを作成します。次のコントロールプレーンノードの Butane 設定の例では、ディスク暗号化用に **master-storage.bu** という名前のファイルを作成します。

```
variant: openshift
version: 4.16.0
metadata:
  name: master-storage
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  luks:
    - clevis:
        tang:
          - thumbprint: QcPr_NHFJammnRCA3fFMVdNBwjs
            url: http://clevis.example.com:7500
        options: (1)
          - --cipher
          - aes-cbc-essiv:sha256
      device: /dev/disk/by-partlabel/root (2)
      label: luks-root
      name: root
      wipe_volume: true
  filesystems:
    - device: /dev/mapper/root
      format: xfs
      label: root
      wipe_filesystem: true
openshift:
  fips: true (3)
```

(1) 暗号オプションは、FIPS モードが有効な場合にのみ必要です。FIPS が無効になっている場合は、エントリーを省略します。

2 DASD タイプのディスクにインストールする場合は、**device:/dev/disk/by-label/root** に置き換えます。

3 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

2. 次のコマンドを実行して、マシンを起動するためのカスタマイズされた initramfs ファイルを作成します。

```
$ coreos-installer pxe customize \
    /root/rhcos-bootfiles/rhcos-<release>-live-initramfs.s390x.img \
    --dest-device /dev/disk/by-id/scsi-<serial_number> --dest-karg-append \
    ip=<ip_address>::<gateway_ip>:<subnet_mask>::<network_device>:none \
    --dest-karg-append nameserver=<nameserver_ip> \
    --dest-karg-append rd.neednet=1 -o \
    /root/rhcos-bootfiles/<node_name>-initramfs.s390x.img
```

注記

最初のブートの前に、クラスター内の各ノードの initramfs をカスタマイズし、PXE カーネルパラメーターを追加する必要があります。

3. **ignition.platform.id=metal** および **ignition.firstboot** を含むパラメーターファイルを作成します。

コントロールプレーンマシンのカーネルパラメーターファイルの例:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/dasda \ 1
ignition.firstboot ignition.platform.id=metal \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \ 2
coreos.inst.ignition_url=http://<http_server>/master.ign \ 3
ip=10.19.17.2::10.19.17.1:255.255.255.0::enbdd0:none nameserver=10.19.17.1 \
zfcp.allow_lun_scan=0 \ 4
rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
rd.zfcp=0.0.5677,0x600606680g7f0056,0x034F000000000000 \ 5
```

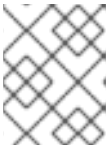
1 DASD タイプのディスクにインストールする場合は、**coreos.inst.install_dev=/dev/dasda** を追加します。FCP タイプのディスクの場合は、この値を省略します。

2 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

3 Ignition 設定ファイルの場所を指定します。**master.ign** または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

4 FCP タイプのディスクにインストールする場合は、**zfcp.allow_lun_scan=0** を追加します。DASD タイプのディスクの場合は、この値を省略します。

5 DASD タイプのディスクにインストールする場合は、**rd.dasd=0.0.3490** に置き換えて DASD デバイスを指定します。



注記

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

関連情報

- Butane でのマシン設定の作成

18.2.13. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Z® インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を z/VM ゲスト仮想マシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS z/VM ゲスト仮想マシンの再起動後に自動的に開始されます。

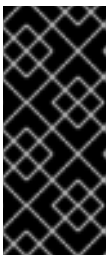
マシンを作成するには、以下の手順を実行します。

前提条件

- 作成するマシンがアクセスできるプロビジョニングマシンで稼働している HTTP または HTTPS サーバー。
- セキュアブートを有効にする場合は、適切な Product Signing Key を取得し、IBM ドキュメントの Secure boot on IBM Z and IBM LinuxONE を確認した。

手順

1. プロビジョニングマシンで Linux にログインします。
2. RHCOS イメージミラー から Red Hat Enterprise Linux CoreOS (RHCOS) カーネル、 initramfs および rootfs ファイルを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な kernel、initramfs、および rootfs アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**

- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

注記

rootfs イメージは FCP および DASD の場合と同じです。

3. パラメーターファイルを作成します。以下のパラメーターは特定の仮想マシンに固有のものです。
   - **ip=** には、以下の 7 つのエントリーを指定します。
     i. マシンの IP アドレス。
     ii. 空の文字列。
     iii. ゲートウェイ。
     iv. ネットマスク。
     v. **hostname.domainname** 形式のマシンホストおよびドメイン名。この値を省略して、RHCOS に決定させるようにします。
     vi. ネットワークインターフェイス名。この値を省略して、RHCOS に決定させるようにします。
     vii. 静的 IP アドレスを使用する場合、 **none** を指定します。
   - **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
   - **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
   - オプション: セキュアブートを有効にするには、**coreos.inst.secure_ipl** を追加します。
   - DASD タイプのディスクへのインストールには、以下のタスクを実行します。
     i. **coreos.inst.install_dev=** には、**/dev/dasda** を指定します。
     ii. **rd.dasd=** を使用して、RHCOS がインストールされる DASD を指定します。
     iii. その他のパラメーターはすべて変更しません。
     ブートストラップマシンのパラメーターファイルのサンプル **bootstrap-0.parm**:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/dasda \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.
<architecture>.img \ 1
coreos.inst.ignition_url=http://<http_server>/bootstrap.ign \ 2
coreos.inst.secure_ipl \ 3
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
```

```
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.dasd=0.0.3490
```

1. 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
2. Ignition 設定ファイルの場所を指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
3. オプション: セキュアブートを有効にするには、**coreos.inst.secure_ipl** を追加します。

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

- FCP タイプのディスクへのインストールには、以下のタスクを実行します。

  i. **rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して RHCOS がインストールされる FCP ディスクを指定します。マルチパスの場合、それぞれの追加のステップについてこのステップを繰り返します。

  注記

  複数のパスを使用してインストールする場合は、問題が発生する可能性があるため、後でではなくインストールの直後にマルチパスを有効にする必要があります。

  ii. インストールデバイスを **coreos.inst.install_dev=/dev/disk/by-id/scsi-<serial_number>** として設定します。

  注記

  追加の LUN が NPIV で設定される場合は、FCP に **zfcp.allow_lun_scan=0** が必要です。CSI ドライバーを使用するために **zfcp.allow_lun_scan=1** を有効にする必要がある場合などには、各ノードが別のノードのブートパーティションにアクセスできないように NPIV を設定する必要があります。

  iii. その他のパラメーターはすべて変更しません。

  重要

  マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

  以下は、マルチパスが設定されたコンピュートノードのパラメーターファイルのサンプル **worker-1.parm** です。

```
rd.neednet=1 \
console=ttysclp0 \
```

```
coreos.inst.install_dev=/dev/disk/by-id/scsi-<serial_number> \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.
<architecture>.img \
coreos.inst.ignition_url=http://<http_server>/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

4. FTP などを使用し、initramfs、kernel、パラメーターファイル、および RHCOS イメージを z/VM に転送します。FTP でファイルを転送し、仮想リーダーから起動する方法については、Z/VM 環境へのインストール を参照してください。

5. ブートストラップノードになる z/VM ゲスト仮想マシンの仮想リーダーに対してファイルの punch を実行します。
IBM ドキュメントの PUNCH を参照してください。

   ヒント

   CP PUNCH コマンドを使用するか、Linux を使用している場合は、**vmur** コマンドを使用して 2 つの z/VM ゲスト仮想マシン間でファイルを転送できます。

6. ブートストラップマシンで CMS にログインします。

7. リーダーからブートストラップマシンに対して IPL を実行します。

   ```
   $ ipl c
   ```

   IBM ドキュメントの IPL を参照してください。

8. クラスター内の他のマシンについてこの手順を繰り返します。

18.2.13.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

18.2.13.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**

- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1つまたは複数のネットワークを設定する場合、1つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
<name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、**options** はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードを使用する場合の問題を回避するために、常にアクティブバックアップモードで **fail_over_mac=1** オプションを設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

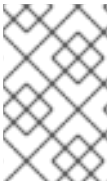
```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1**、**em2**) のコンマ区切りリストを表します。



注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

18.2.14. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

18.2.15. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

18.2.16. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

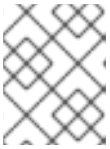
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
```

```
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

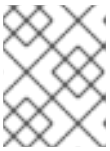
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.29.4
   master-1  Ready     master  73m  v1.29.4
   master-2  Ready     master  74m  v1.29.4
   worker-0  Ready     worker  11m  v1.29.4
   worker-1  Ready     worker  11m  v1.29.4
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

18.2.17. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

18.2.17.1. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

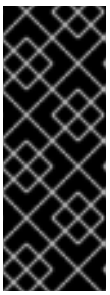
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

18.2.17.1.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z® にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。



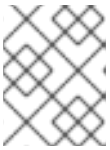
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

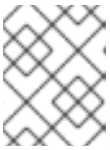
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION           AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16              True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

18.2.17.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

18.2.18. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
```

```
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

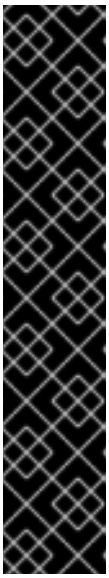
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver             apiserver-z25h4                             1/1     Running    0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
Running    0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

検証

OpenShift Container Platform のブートストラッププロセス中にセキュアブートを有効にした場合は、次の検証手順が必要です。

1. 次のコマンドを実行してノードをデバッグします。

```
$ oc debug node/<node_name>
chroot /host
```

2. 次のコマンドを実行して、セキュアブートが有効になっていることを確認します。

```
$ cat /sys/firmware/ipl/secure
```

出力例

```
1 1
```

1 セキュアブートが有効になっている場合、値は **1** です。有効になっていない場合は **0** です。

18.2.19. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。
- How to generate SOSREPORT within OpenShift4 nodes without SSH

18.2.20. 次のステップ

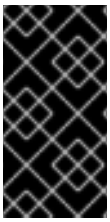
- RHCOS でカーネル引数を使用してマルチパスを有効化 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

18.3. ネットワークが制限された環境での Z/VM のあるクラスターの IBM Z および IBM LINUXONE へのインストール

OpenShift Container Platform バージョン 4.16 では、制限されたネットワーク内で、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャーに、クラスターをインストールできます。

注記

このドキュメントは IBM Z® のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

18.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ネットワークが制限された環境でインストールのミラーレジストリーを作成 し、お使いの OpenShift Container Platform のバージョンの **imageContentSources** データを取得している。

- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

重要

インストールメディアにアクセスできるマシンからインストール手順が実行されるようにします。

- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。

- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

18.3.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

18.3.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。

- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

18.3.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

18.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

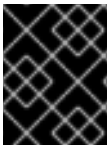
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

18.3.4.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表18.21 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

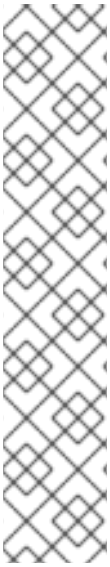
18.3.4.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表18.22 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
Compute	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

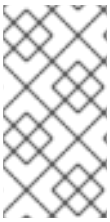
18.3.4.3. 最小の IBM Z システム環境

OpenShift Container Platform バージョン 4.16 は、以下の IBM® ハードウェアにインストールできます。

- IBM® z16 (すべてのモデル)、IBM® z15 (すべてのモデル)、IBM® z14 (すべてのモデル)
- IBM® LinuxONE 4 (すべてのモデル)、IBM® LinuxONE III (すべてのモデル)、IBM® LinuxONE Empire II、IBM® LinuxONE Rockhopper II

ハードウェア要件

- クラスターごとに、SMT2 対応の 6 つの Integrated Facilities for Linux (IFL) に相当します。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。



注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z® の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。

重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR のウェイト管理、エンタイトルメント、および CPU 共有が重要な役割を果たします。

オペレーティングシステム要件

- z/VM 7.2 以降の 1 インスタンス

z/VM インスタンスで、以下をセットアップします。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Z ネットワーク接続の要件
IBM Z® の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM vSwitch のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

ディスクストレージ

- FICON 接続のディスクストレージ (DASD)。これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。

- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB

- OpenShift Container Platform コンピュートマシン用に 8 GB

- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

18.3.4.4. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。

- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

- HiperSockets。ノードに直接割り当てたものか、z/VM ゲストに対して透過的になるように z/VM VSWITCH でブリッジしてノードに割り当てたもの。HiperSockets をノードに直接接続するには、RHEL 8 ゲストを介して外部ネットワークへのゲートウェイをセットアップし、HiperSockets ネットワークにブリッジする必要があります。

オペレーティングシステム要件

- 高可用性を確保する場合は z/VM 7.2 以降の 2 または 3 インスタンス

z/VM インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用に 3 ゲスト仮想マシン (z/VM インスタンスごとに 1 つ)

- OpenShift Container Platform コンピュートマシン用に 6 以上のゲスト仮想マシン (z/VM インスタンス全体に分散)

- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

- オーバーコミット環境で必須コンポーネントの可用性を確保するには、CP コマンドの **SET SHARE** を使用してコントロールプレーンの優先度を引き上げます。インフラストラクチャーノードが存在する場合は、同じ操作を行います。IBM® ドキュメントの SET SHARE を参照してください。

IBM Z ネットワーク接続の要件
IBM Z® の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター

- z/VM vSwitch のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

ディスクストレージ

- FICON 接続のディスクストレージ (DASD)。これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

18.3.4.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

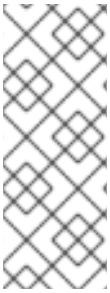
- IBM® ドキュメントの Bridging a HiperSockets LAN with a z/VM Virtual Switch を参照してください。
- パフォーマンスの最適化については、Scaling HyperPAV alias devices on Linux guests on z/VM を参照してください。
- LPAR のウェイト管理とエンタイトルメントについては、Topics in LPAR Performance を参照してください。
- IBM Z® および IBM® LinuxONE 環境に推奨されるホストプラクティス

18.3.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。



注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

18.3.4.6.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

18.3.4.6.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表18.23 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve

プロトコル	ポート	説明
	9000-9999	ポート **9100-9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000-32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表18.24 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表18.25 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

18.3.4.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード

- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

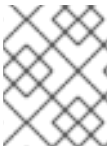
DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表18.26 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

18.3.4.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例18.4 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例18.5 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

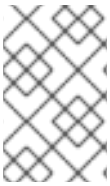
7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

18.3.4.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.27 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

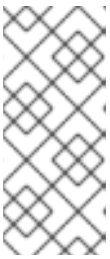
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.28 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

18.3.4.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例18.6 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
```

```
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 1
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup 2
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 3
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 4
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 5
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 6
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

18.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

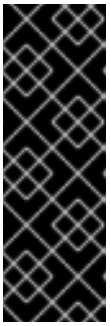
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。



重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

5. クラスターに必要な DNS インフラストラクチャーを設定します。

    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

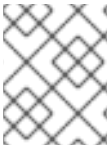
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

6. DNS 設定を検証します。

    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

18.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   注記

   出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

   random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

18.3.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

18.3.8. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

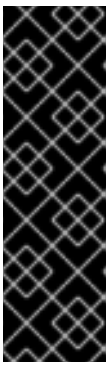
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

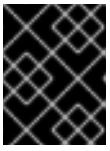
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Z® のインストール設定パラメーター

18.3.8.1. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

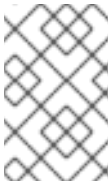
```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture: s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
```

```
  architecture: s390x
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OVNKubernetes 11
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
additionalTrustBundle: | 17
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 18
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が OpenShift Container Platform ノードで利用できない場合、**hyperthreading** パラメーターは影響を受けません。

重要

OpenShift Container Platform ノードまたは **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプ

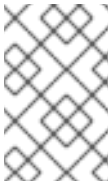
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Z® インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパス

し、代わりに RHCOS で提供される暗号モジュールを使用します。

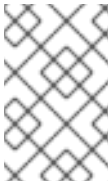
重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

18 リポジトリーのミラーリングに使用したコマンドの出力に従って、**imageContentSources** セクションを指定します。

重要

- **oc adm release mirror** コマンドを使用する場合は、**imageContentSources** セクションの出力を使用します。
- **oc mirror** コマンドを使用する場合は、コマンドの実行によって生成される **ImageContentSourcePolicy** ファイルの **repositoryDigestMirrors** セクションを使用します。
- **ImageContentSourcePolicy** は非推奨になりました。詳細は、**イメージレジストリーリポジトリーミラーリングの設定** を参照してください。

18.3.8.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

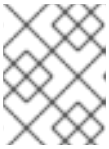
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

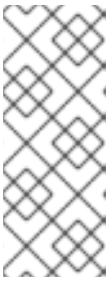
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** で



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

18.3.8.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定される最小の 3 つのノードクラスターにゼロコンピューティングマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

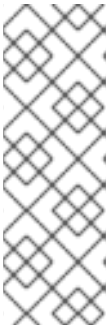
前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```



注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

18.3.9. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

18.3.9.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表18.29 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表18.30 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表18.31 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。

フィールド	型	説明
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。  **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表18.32 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表18.33 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表18.34 **policyAuditConfig** オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表18.35 **gatewayConfig** オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表18.36 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表18.37 **gatewayConfig.ipv6** object

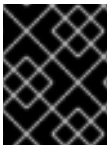
フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表18.38 **ipsecConfig** オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```



重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)

kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表18.39 **kubeProxyConfig** オブジェクト

フィールド	型	説明

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。 **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

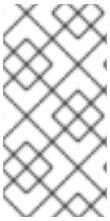
18.3.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.3.11. IBM Z または IBM& LinuxONE 環境での静的 IP を使用した NBDE の設定

IBM Z® または IBM® LinuxONE 環境で NBDE ディスク暗号化を有効にするには、追加の手順が必要です。このセクションで詳しく説明します。

前提条件

- External Tang Server がセットアップされました。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。
- **butane** ユーティリティーをインストールしている。
- Butane でマシン設定を作成する手順を確認している。

手順

1. コントロールプレーンとコンピュートノードの Butane 設定ファイルを作成します。
次のコントロールプレーンノードの Butane 設定の例では、ディスク暗号化用に **master-storage.bu** という名前のファイルを作成します。

```
variant: openshift
version: 4.16.0
metadata:
  name: master-storage
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  luks:
    - clevis:
        tang:
          - thumbprint: QcPr_NHFJammnRCA3fFMVdNBwjs
            url: http://clevis.example.com:7500
        options: 1
          - --cipher
          - aes-cbc-essiv:sha256
      device: /dev/disk/by-partlabel/root 2
```

```
      label: luks-root
      name: root
      wipe_volume: true
  filesystems:
    - device: /dev/mapper/root
      format: xfs
      label: root
      wipe_filesystem: true
openshift:
  fips: true 3
```

1. 暗号オプションは、FIPS モードが有効な場合にのみ必要です。FIPS が無効になっている場合は、エントリーを省略します。
2. DASD タイプのディスクにインストールする場合は、**device:/dev/disk/by-label/root** に置き換えます。
3. FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

2. 次のコマンドを実行して、マシンを起動するためのカスタマイズされた initramfs ファイルを作成します。

```
$ coreos-installer pxe customize \
    /root/rhcos-bootfiles/rhcos-<release>-live-initramfs.s390x.img \
    --dest-device /dev/disk/by-id/scsi-<serial_number> --dest-karg-append \
    ip=<ip_address>::<gateway_ip>:<subnet_mask>::<network_device>:none \
    --dest-karg-append nameserver=<nameserver_ip> \
    --dest-karg-append rd.neednet=1 -o \
    /root/rhcos-bootfiles/<node_name>-initramfs.s390x.img
```

注記

最初のブートの前に、クラスター内の各ノードの initramfs をカスタマイズし、PXE カーネルパラメーターを追加する必要があります。

3. **ignition.platform.id=metal** および **ignition.firstboot** を含むパラメーターファイルを作成します。

コントロールプレーンマシンのカーネルパラメーターファイルの例:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/dasda \1
ignition.firstboot ignition.platform.id=metal \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \2
coreos.inst.ignition_url=http://<http_server>/master.ign \3
ip=10.19.17.2::10.19.17.1:255.255.255.0::enbdd0:none nameserver=10.19.17.1 \
zfcp.allow_lun_scan=0 \4
rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
rd.zfcp=0.0.5677,0x600606680g7f0056,0x034F000000000000 \5
```

1. DASD タイプのディスクにインストールする場合は、**coreos.inst.install_dev=/dev/dasda** を追加します。FCP タイプのディスクの場合は、この値を省略します。
2. 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
3. Ignition 設定ファイルの場所を指定します。**master.ign** または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
4. FCP タイプのディスクにインストールする場合は、**zfcp.allow_lun_scan=0** を追加します。DASD タイプのディスクの場合は、この値を省略します。
5. DASD タイプのディスクにインストールする場合は、**rd.dasd=0.0.3490** に置き換えて DASD デバイスを指定します。

注記

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

関連情報

- Butane でのマシン設定の作成

18.3.12. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Z® インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を z/VM ゲスト仮想マシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS z/VM ゲスト仮想マシンの再起動後に自動的に開始されます。

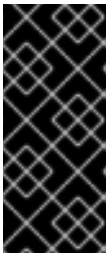
マシンを作成するには、以下の手順を実行します。

前提条件

- 作成するマシンがアクセスできるプロビジョニングマシンで稼働している HTTP または HTTPS サーバー。
- セキュアブートを有効にする場合は、適切な Product Signing Key を取得し、IBM ドキュメントの Secure boot on IBM Z and IBM LinuxONE を確認した。

手順

1. プロビジョニングマシンで Linux にログインします。
2. RHCOS イメージミラー から Red Hat Enterprise Linux CoreOS (RHCOS) カーネル、 initramfs および rootfs ファイルを取得します。

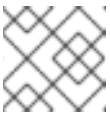


重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な kernel、initramfs、および rootfs アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**



注記

rootfs イメージは FCP および DASD の場合と同じです。

3. パラメーターファイルを作成します。以下のパラメーターは特定の仮想マシンに固有のものです。
   - **ip=** には、以下の 7 つのエントリーを指定します。
     i. マシンの IP アドレス。
     ii. 空の文字列。
     iii. ゲートウェイ。
     iv. ネットマスク。
     v. **hostname.domainname** 形式のマシンホストおよびドメイン名。この値を省略して、RHCOS に決定させるようにします。
     vi. ネットワークインターフェイス名。この値を省略して、RHCOS に決定させるようにします。
     vii. 静的 IP アドレスを使用する場合、 **none** を指定します。
   - **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
   - **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
   - オプション: セキュアブートを有効にするには、**coreos.inst.secure_ipl** を追加します。
   - DASD タイプのディスクへのインストールには、以下のタスクを実行します。
     i. **coreos.inst.install_dev=** には、**/dev/dasda** を指定します。
     ii. **rd.dasd=** を使用して、RHCOS がインストールされる DASD を指定します。

iii. その他のパラメーターはすべて変更しません。
ブートストラップマシンのパラメーターファイルのサンプル **bootstrap-0.parm**:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/dasda \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.
<architecture>.img \ 1
coreos.inst.ignition_url=http://<http_server>/bootstrap.ign \ 2
coreos.inst.secure_ipl \ 3
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.dasd=0.0.3490
```

1 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

2 Ignition 設定ファイルの場所を指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

3 オプション: セキュアブートを有効にするには、**coreos.inst.secure_ipl** を追加します。

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

- FCP タイプのディスクへのインストールには、以下のタスクを実行します。

i. **rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して RHCOS がインストールされる FCP ディスクを指定します。マルチパスの場合、それぞれの追加のステップについてこのステップを繰り返します。

注記

複数のパスを使用してインストールする場合は、問題が発生する可能性があるため、後でではなくインストールの直後にマルチパスを有効にする必要があります。

ii. インストールデバイスを **coreos.inst.install_dev=/dev/disk/by-id/scsi-<serial_number>** として設定します。

注記

追加の LUN が NPIV で設定される場合は、FCP に **zfcp.allow_lun_scan=0** が必要です。CSI ドライバーを使用するために **zfcp.allow_lun_scan=1** を有効にする必要がある場合などには、各ノードが別のノードのブートパーティションにアクセスできないように NPIV を設定する必要があります。

iii. その他のパラメーターはすべて変更しません。

重要

マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

以下は、マルチパスが設定されたコンピュートノードのパラメーターファイルのサンプル **worker-1.parm** です。

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/disk/by-id/scsi-<serial_number> \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.
<architecture>.img \
coreos.inst.ignition_url=http://<http_server>/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

4. FTP などを使用し、initramfs、kernel、パラメーターファイル、および RHCOS イメージを z/VM に転送します。FTP でファイルを転送し、仮想リーダーから起動する方法については、Z/VM 環境へのインストール を参照してください。

5. ブートストラップノードになる z/VM ゲスト仮想マシンの仮想リーダーに対してファイルの punch を実行します。
IBM ドキュメントの PUNCH を参照してください。

   ヒント

   CP PUNCH コマンドを使用するか、Linux を使用している場合は、**vmur** コマンドを使用して 2 つの z/VM ゲスト仮想マシン間でファイルを転送できます。

6. ブートストラップマシンで CMS にログインします。

7. リーダーからブートストラップマシンに対して IPL を実行します。

   ```
   $ ipl c
   ```

   IBM ドキュメントの IPL を参照してください。

8. クラスター内の他のマシンについてこの手順を繰り返します。

18.3.12.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマ

ンドラインのオプションを説明します。

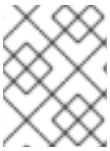
18.3.12.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

  ```
  ip=::10.10.10.254::::
  ```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

  ```
  rd.route=20.20.20.0/24:20.20.20.254:enp2s0
  ```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

  ```
  ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
  vlan=enp2s0.100:enp2s0
  ```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

  ```
  ip=enp2s0.100:dhcp
  vlan=enp2s0.100:enp2s0
  ```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、**options** はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:em1,em2:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードを使用する場合の問題を回避するために、常にアクティブバックアップモードで **fail_over_mac=1** オプションを設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1**、**em2**) のコンマ区切りリストを表します。

注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

18.3.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。

- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

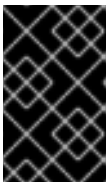
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

18.3.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

18.3.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
master-2 Ready    master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

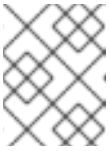
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

18.3.16. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

18.3.16.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

18.3.16.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

18.3.16.2.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z® にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

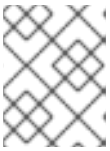
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

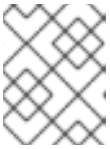
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION        AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry    4.16           True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

18.3.16.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

18.3.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

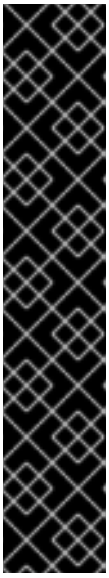
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver             apiserver-z25h4                             1/1     Running    0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
Running    0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace>
```
(1)

(1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

4. Cluster registration ページでクラスターを登録します。

検証

OpenShift Container Platform のブートストラッププロセス中にセキュアブートを有効にした場合は、次の検証手順が必要です。

1. 次のコマンドを実行してノードをデバッグします。

```
$ oc debug node/<node_name>
chroot /host
```

2. 次のコマンドを実行して、セキュアブートが有効になっていることを確認します。

```
$ cat /sys/firmware/ipl/secure
```

出力例

```
1
```
(1)

(1) セキュアブートが有効になっている場合、値は **1** です。有効になっていない場合は **0** です。

関連情報

- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH

18.3.18. 次のステップ

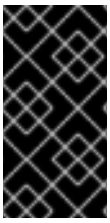
- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。

18.4. RHEL KVM を使用したクラスターの IBM Z および IBM LINUXONE へのインストール

OpenShift Container Platform バージョン 4.16 では、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャーに、クラスターをインストールできます。

注記

このドキュメントは IBM Z® のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

18.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

  注記

  プロキシーを設定する場合は、このサイト一覧も確認してください。

- 論理パーティション (LPAR) でホストされ、RHEL 8.6 以降をベースとする RHEL Kernel Virtual Machine (KVM) システムをプロビジョニングしている。Red Hat Enterprise Linux 8 and 9 Life Cycle を参照してください。

18.4.2. OpenShift Container Platform のインターネットアクセス

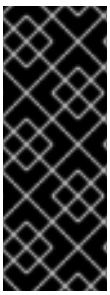
OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用し

ます。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

18.4.3. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

RHEL 8.6 以降をベースとする 1 つ以上の KVM ホストマシン。各 RHEL KVM ホストマシンで libvirt がインストールされ、実行している必要があります。仮想マシンは、各 RHEL KVM ホストマシンでプロビジョニングされます。

18.4.3.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表18.40 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる RHEL インスタンスにコントロールプレーンマシンを分散します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

18.4.3.2. ネットワーク接続の要件

OpenShift Container Platform インストーラーは、すべての Red Hat Enterprise Linux CoreOS (RHCOS) 仮想マシンに必要な Ignition ファイルを作成します。OpenShift Container Platform の自動インストールはブートストラップマシンで実行されます。これは各ノードで OpenShift Container Platform のインストールを開始し、Kubernetes クラスターを起動してから終了します。このブートストラップ時に、仮想マシンには Dynamic Host Configuration Protocol (DHCP) サーバーまたは静的 IP アドレスでネットワーク接続を確立している必要があります。

18.4.3.3. IBM Z ネットワーク接続の要件

RHEL KVM の IBM Z® にインストールするには、以下が必要です。

- OSA または RoCE ネットワークアダプターで設定された RHEL KVM ホスト。
- libvirt または MacVTap のブリッジネットワークを使用してネットワークをゲストに接続するように設定されているいずれかの RHEL KVM ホスト。
  仮想ネットワーク接続の種類 を参照してください。

18.4.3.4. ホストマシンのリソース要件

お使いの環境の RHEL KVM ホストは、OpenShift Container Platform 環境用に計画している仮想マシンをホストするために以下の要件を満たす必要があります。仮想化の使用開始 を参照してください。

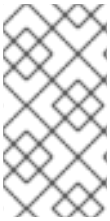
OpenShift Container Platform バージョン 4.16 は、以下の IBM® ハードウェアにインストールできます。

- IBM® z16 (すべてのモデル)、IBM® z15 (すべてのモデル)、IBM® z14 (すべてのモデル)
- IBM® LinuxONE 4 (すべてのモデル)、IBM® LinuxONE III (すべてのモデル)、IBM® LinuxONE Empire II、IBM® LinuxONE Rockhopper II

18.4.3.5. 最小の IBM Z システム環境

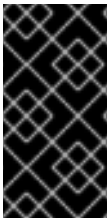
ハードウェア要件

- クラスターごとに、SMT2 対応の 6 つの Integrated Facilities for Linux (IFL) に相当します。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。



注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z® の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。



重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR のウェイト管理、エンタイトルメント、および CPU 共有が重要な役割を果たします。

オペレーティングシステム要件

- libvirt で管理される、KVM を使用する RHEL 8.6 以降で実行する 1 つの LPAR。

RHEL KVM ホストで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

18.4.3.6. 最小リソース要件

それぞれのクラスターの仮想マシンは、以下の最小要件を満たしている必要があります。

仮想マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
Compute	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

18.4.3.7. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

オペレーティングシステム要件

- 高可用性が必要な場合は、libvirt で管理される、KVM を使用する RHEL 8.6 以降で実行する 2 または 3 つの LPAR。

RHEL KVM ホストで、以下を設定します。

- OpenShift Container Platform コンピュートプレーンマシン用に 3 つのゲスト仮想マシン (RHEL KVM ホストマシン全体に分散)
- OpenShift Container Platform コンピュートマシン用に 6 つ以上のゲスト仮想マシン (RHEL KVM ホストマシン全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン
- オーバーコミット環境で必須コンポーネントの可用性を確保するには、**cpu_shares** を使用してコントロールプレーンの優先度を引き上げます。インフラストラクチャーノードが存在する場合は、同じ操作を行います。IBM® ドキュメントの schedinfo を参照してください。

18.4.3.8. 優先されるリソース要件

各クラスターの仮想マシンについての優先される要件は以下の通りです。

仮想マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	8	16 GB	120 GB
Compute	RHCOS	6	8 GB	120 GB

18.4.3.9. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- IBM Z® および IBM® LinuxONE 環境に推奨されるホストプラクティス

18.4.3.10. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

18.4.3.10.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

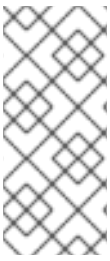
18.4.3.10.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。



注記

RHEL KVM ホストは、libvirt または MacVTap のブリッジネットワークを使用して、ネットワークを仮想マシンに接続するように設定される必要があります。仮想マシンには、RHEL KVM ホストに接続されているネットワークへのアクセスがある必要があります。KVM 内の仮想ネットワーク (ネットワークアドレス変換 (NAT) など) はサポートされる設定ではありません。

表18.41 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表18.42 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表18.43 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP)

サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

18.4.3.11. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

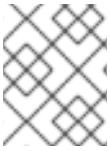
DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表18.44 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

18.4.3.11.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例18.7 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
```

```
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例18.8 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
```

```
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

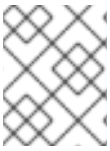
1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

18.4.3.12. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

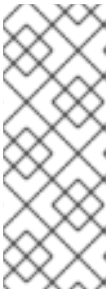
重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.45 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.46 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

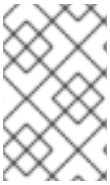
注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

18.4.3.12.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例18.9 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
```

```
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 1
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup 2
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 3
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 4
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 5
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 6
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

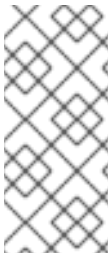
2 4

ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

18.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

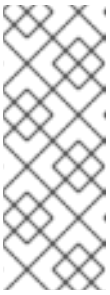
- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。

a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。



注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. Red Hat Enterprise Linux CoreOS (RHCOS) の 高速インストールまたは Red Hat Enterprise Linux CoreOS (RHCOS) のフルインストールのいずれかの実行を選択します。フルインストールでは、HTTP または HTTPS サーバーを設定し、Ignition ファイルを提供し、イメージをクラスターノードにインストールする必要があります。高速インストールの場合、HTTP または HTTPS サーバーは必要はありませんが、DHCP サーバーが必要です。高速インストール: Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成およびフルインストール: Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成のセクションを参照してください。

3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

5. クラスターに必要な DNS インフラストラクチャーを設定します。

a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

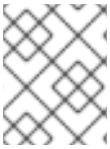
b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

6. DNS 設定を検証します。

a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

18.4.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain>
```
(1)

–

1. **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

出力例

```
api.ocp4.example.com.  604800 IN A 192.168.1.5
```

b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

出力例

```
api-int.ocp4.example.com.  604800 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

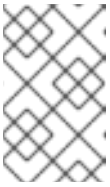
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

18.4.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各

ノードの **core** ユーザーの ~/**.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

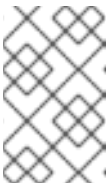
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

18.4.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

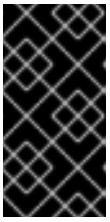
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。

3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

18.4.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

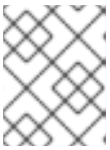
```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。

3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。

5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

18.4.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

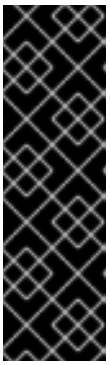
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

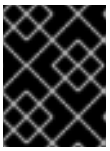
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Z® のインストール設定パラメーター

18.4.9.1. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture: s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture: s390x
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths": ...}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行は

ハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

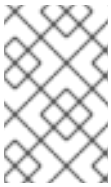
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が OpenShift Container Platform ノードで利用できない場合、**hyperthreading** パラメーターは影響を受けません。

重要

OpenShift Container Platform ノードまたは **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



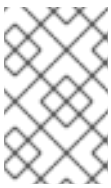
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを

提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Z® インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

18.4.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする

trusted-ca-bundle 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

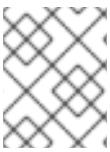
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

18.4.9.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定される最小の 3 つのノードクラスターにゼロコンピューティングマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

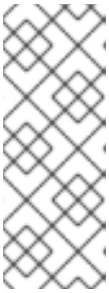
手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。



注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

18.4.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

18.4.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表18.47 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表18.48 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表18.49 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。

フィールド	型	説明
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。  **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表18.50 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。

フィールド	型	説明
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表18.51 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表18.52 **policyAuditConfig オブジェクト**

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。

フィールド	型	説明
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表18.53 **gatewayConfig** オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

フィールド	型	説明
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表18.54 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表18.55 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表18.56 **ipsecConfig** オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

–

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

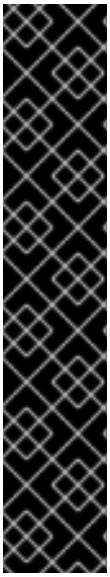
表18.57 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: `kubeProxyConfig:` `  proxyArguments:` `  iptables-min-sync-period:` `  - 0s`

18.4.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

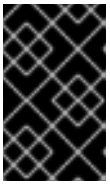
1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.4.12. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform をプロビジョニングする IBM Z® インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を Red Hat Enterprise Linux (RHEL) ゲスト仮想マシンとしてインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

事前にパッケージ化された QEMU コピーオンライト (QCOW2) ディスクイメージを使用する RHCOS の高速インストールを実行できます。または、新規の QCOW2 ディスクイメージでフルインストールを実行できます。

システムのセキュリティーをさらに強化するために、ファストトラックインストールに進む前に、オプションで IBM® Secure Execution を使用して RHCOS をインストールできます。

18.4.12.1. IBM Secure Execution を使用した RHCOS のインストール

IBM® Secure Execution を使用して RHCOS をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

前提条件

- IBM® z15 以降、または IBM® LinuxONE III 以降。
- Red Hat Enterprise Linux (RHEL) 8 以降
- ブートストラップ Ignition ファイルがあります。ファイルは保護されていないため、他のユーザーが表示および編集できます。
- インストール後にブートイメージが変更されていないことを確認しました。
- すべてのノードを IBM® Secure Execution ゲストとして実行する必要があります。

手順

1. RHEL KVM ホストを準備して、IBM® Secure Execution をサポートします。
   - デフォルトでは、KVM ホストは IBM® Secure Execution モードのゲストをサポートしません。ゲストを IBM® Secure Execution モードでサポートするには、KVM ホストを LPAR モードで起動し、カーネルパラメーターを **prot_virt=1** に指定する必要があります。RHEL 8 で **prot_virt=1** を有効にするには、次の手順に従います。
     a. **/boot/loader/entries/** に移動して、ブートローダー設定ファイル ***.conf** を変更します。
     b. カーネルコマンドラインパラメーター **prot_virt=1** を追加します。
     c. **zipl** コマンドを実行し、システムを再起動します。
     IBM® Secure Execution for Linux のサポートを使用して正常に始動する KVM ホストは、次のカーネルメッセージを発行します。

     ```
     prot_virt: Reserving <amount>MB as ultravisor base storage.
     ```

     d. KVM ホストが IBM® Secure Execution をサポートするようになったことを確認するには、次のコマンドを実行します。

     ```
     # cat /sys/firmware/uv/prot_virt_host
     ```

     出力例

     ```
     1
     ```

     環境がセキュアホストの環境と一致していると検出された Linux インスタンスの場合、この属性の値は 1 です。他のインスタンスの場合、値は 0 です。
2. Ignition を介して KVM ゲストにホストキーを追加します。
   最初の起動時に、RHCOS はホストキーを探して、それを使用して自身を再暗号化します。

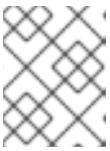
RHCOS は、**/etc/se-hostkeys** ディレクトリーで **ibm-z-hostkey-** で始まるファイルを検索します。クラスターが実行されている各マシンのすべてのホストキーは、管理者がディレクトリーにロードする必要があります。最初の起動後、他のマシンで VM を実行することはできません。

注記

安全なシステムで Ignition ファイルを準備する必要があります。例えば、別の IBM® Secure Execution ゲストです。

以下に例を示します。

```
{
  "ignition": { "version": "3.0.0" },
  "storage": {
    "files": [
      {
        "path": "/etc/se-hostkeys/ibm-z-hostkey-<your-hostkey>.crt",
        "contents": {
          "source": "data:;base64,<base64 encoded hostkey document>"
        },
        "mode": 420
      },
      {
        "path": "/etc/se-hostkeys/ibm-z-hostkey-<your-hostkey>.crt",
        "contents": {
          "source": "data:;base64,<base64 encoded hostkey document>"
        },
        "mode": 420
      }
    ]
  }
}
```



注記

ノードを複数の IBM Z® マシンで実行できるようにする場合は、必要な数のホストキーを追加できます。

3. Base64 でエンコードされた文字列を生成するには、次のコマンドを実行します。

```
base64 <your-hostkey>.crt
```

IBM® Secure Execution を実行していないゲストと比較すると、イグニションフェーズの前にイメージ全体がランダムに生成された LUKS パスフレーズで暗号化されるため、マシンの最初のブートに時間がかかります。

4. Ignition 保護の追加
Ignition 設定ファイルに保存されているシークレットが読み取られたり変更されたりしないようにするには、Ignition 設定ファイルを暗号化する必要があります。

注記

必要なセキュリティーを実現するために、IBM® Secure Execution の実行時は、Ignition ロギングとローカルログインがデフォルトで無効になります。

a. 次のコマンドを実行して、**secex-qemu.qcow2** イメージの公開 GPG キーを取得し、そのキーを使用して Ignition 設定を暗号化します。

```
gpg --recipient-file /path/to/ignition.gpg.pub --yes --output /path/to/config.ign.gpg --
verbose --armor --encrypt /path/to/config.ign
```

5. RHCOS の高速インストールに従って、IBM® Secure Execution QCOW イメージを使用してノードをインストールします。

注記

仮想マシンを起動する前に、**serial=ignition** を **serial=ignition_crypted** に置き換え、**launchSecurity** パラメーターを追加します。

検証

RHCOS の高速インストールが完了し、最初の起動時に Ignition が実行したら、復号が成功したかどうかを確認します。

- 復号が成功すると、次の例のような出力が期待できます。

出力例

```
[    2.801433] systemd[1]: Starting coreos-ignition-setup-user.service - CoreOS Ignition User
Config Setup...

[    2.803959] coreos-secex-ignition-decrypt[731]: gpg: key <key_name>: public key "Secure
Execution (secex) 38.20230323.dev.0" imported
[    2.808874] coreos-secex-ignition-decrypt[740]: gpg: encrypted with rsa4096 key, ID
<key_name>, created <yyyy-mm-dd>
[  OK  ] Finished coreos-secex-igni…S Secex Ignition Config Decryptor.
```

- 復号nい失敗した場合は、次の例のような出力が予想されます。

出力例

```
Starting coreos-ignition-s…reOS Ignition User Config Setup...
[    2.863675] coreos-secex-ignition-decrypt[729]: gpg: key <key_name>: public key "Secure
Execution (secex) 38.20230323.dev.0" imported
[    2.869178] coreos-secex-ignition-decrypt[738]: gpg: encrypted with RSA key, ID
<key_name>
[    2.870347] coreos-secex-ignition-decrypt[738]: gpg: public key decryption failed: No secret
key
[    2.870371] coreos-secex-ignition-decrypt[738]: gpg: decryption failed: No secret key
```

関連情報

- IBM® Secure Execution for Linux の紹介

- IBM® Secure Execution ホストまたはゲストとしての Linux
- IBM Z での IBM® Secure Execution の設定

18.4.12.2. IBM Z または IBM& LinuxONE 環境での静的 IP を使用した NBDE の設定

IBM Z® または IBM® LinuxONE 環境で NBDE ディスク暗号化を有効にするには、追加の手順が必要です。このセクションで詳しく説明します。

前提条件

- External Tang Server がセットアップされました。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。
- **butane** ユーティリティーをインストールしている。
- Butane でマシン設定を作成する手順を確認している。

手順

1. コントロールプレーンとコンピュートノードの Butane 設定ファイルを作成します。
次のコントロールプレーンノードの Butane 設定の例では、ディスク暗号化用に **master-storage.bu** という名前のファイルを作成します。

```
variant: openshift
version: 4.16.0
metadata:
  name: master-storage
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  luks:
    - clevis:
        tang:
          - thumbprint: QcPr_NHFJammnRCA3fFMVdNBwjs
            url: http://clevis.example.com:7500
        options: (1)
          - --cipher
          - aes-cbc-essiv:sha256
      device: /dev/disk/by-partlabel/root
      label: luks-root
      name: root
      wipe_volume: true
  filesystems:
    - device: /dev/mapper/root
      format: xfs
      label: root
      wipe_filesystem: true
openshift:
  fips: true (2)
```

(1) 暗号オプションは、FIPS モードが有効な場合にのみ必要です。FIPS が無効になっている場合は、エントリーを省略します。

(2) FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にさ

2. 次のコマンドを実行して、マシンを起動するためのカスタマイズされた initramfs ファイルを作成します。

```
$ coreos-installer pxe customize \
    /root/rhcos-bootfiles/rhcos-<release>-live-initramfs.s390x.img \
    --dest-device /dev/disk/by-id/scsi-<serial_number> --dest-karg-append \
    ip=<ip_address>::<gateway_ip>:<subnet_mask>::<network_device>:none \
    --dest-karg-append nameserver=<nameserver_ip> \
    --dest-karg-append rd.neednet=1 -o \
    /root/rhcos-bootfiles/<node_name>-initramfs.s390x.img
```

注記

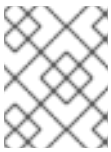
最初のブートの前に、クラスター内の各ノードの initramfs をカスタマイズし、PXE カーネルパラメーターを追加する必要があります。

3. **ignition.platform.id=metal** および **ignition.firstboot** を含むパラメーターファイルを作成します。

```
rd.neednet=1 \
console=ttysclp0 \
ignition.firstboot ignition.platform.id=metal \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \ 1
coreos.inst.ignition_url=http://<http_server>/master.ign \ 2
ip=10.19.17.2::10.19.17.1:255.255.255.0::enbdd0:none nameserver=10.19.17.1 \
zfcp.allow_lun_scan=0 \
rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
rd.zfcp=0.0.5677,0x600606680g7f0056,0x034F000000000000
```

1 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

2 Ignition 設定ファイルの場所を指定します。**master.ign** または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。



注記

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

関連情報

- Butane でのマシン設定の作成

18.4.12.3. 事前にパッケージ化された QCOW2 ディスクイメージを使用した高速インストール

Red Hat Enterprise Linux CoreOS (RHCOS) の高速インストールでマシンを作成するには、事前にパッケージ化された Red Hat Enterprise Linux CoreOS (RHCOS)QEMU コピーオンライト (QCOW2) ディスクイメージをインポートします。

前提条件

- この手順では RHEL KVM ホストと呼ばれる、KVM を使用する RHEL 8.6 以降で実行されている少なくとも 1 つの LPAR。
- KVM/QEMU ハイパーバイザーが RHEL KVM ホストにインストーされている
- ノードのホスト名および逆引き参照を実行できるドメインネームサーバー (DNS)。
- IP アドレスを提供する DHCP サーバー。

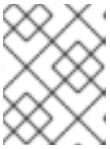
手順

1. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラー ページから RHEL QEMU コピーオンライト (QCOW2) ディスクイメージファイルを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な RHCOS QCOW2 イメージのみを使用します。

2. QCOW2 ディスクイメージおよび Ignition ファイルを RHEL KVM ホストの共通ディレクトリーにダウンロードします。
例: **/var/lib/libvirt/images**



注記

Ignition ファイルは OpenShift Container Platform インストーラーによって生成されます。

3. 各 KVM ゲストノードの QCOW2 ディスクイメージバッキングファイルで、新しいディスクイメージを作成します。

```
$ qemu-img create -f qcow2 -F qcow2 -b /var/lib/libvirt/images/{source_rhcos_qemu}
/var/lib/libvirt/images/{vmname}.qcow2 {size}
```

4. Ignition ファイルと新規ディスクイメージを使用して、新規 KVM ゲストノードを作成します。

```
$ virt-install --noautoconsole \
  --connect qemu:///system \
  --name {vm_name} \
  --memory {memory} \
  --vcpus {vcpus} \
  --disk {disk} \
  --launchSecurity type="s390-pv" \ 1
  --import \
  --network network={network},mac={mac} \
  --disk path={ign_file},format=raw,readonly=on,serial=ignition,startup_policy=optional 2
```

1 IBM® Secure Execution が有効な場合は、**launchSecurity type="s390-pv"** パラメーターを追加します。

2 IBM® Secure Execution が有効な場合は、**serial=ignition** を **Serial=ignition_crypted** に置き換えます。

18.4.12.4. 新規 QCOW2 ディスクイメージへのフルインストール

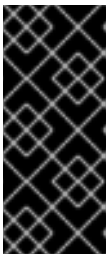
新規 QEMU copy-on-write (QCOW2) ディスクイメージのフルインストールでマシンを作成するには、以下の手順を実施します。

前提条件

- この手順では RHEL KVM ホストと呼ばれる、KVM を使用する RHEL 8.6 以降で実行されている少なくとも 1 つの LPAR。
- KVM/QEMU ハイパーバイザーが RHEL KVM ホストにインストーされている
- ノードのホスト名および逆引き参照を実行できるドメインネームサーバー (DNS)。
- HTTP または HTTPS サーバーが設定されている。

手順

1. Red Hat カスタマーポータルの 製品のダウンロード ページ、または RHCOS イメージミラーページから RHEL カーネル、initramfs、および rootfs ファイルを取得します。



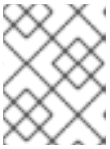
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な RHCOS QCOW2 イメージのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

2. **virt-install** を起動する前に、ダウンロードした RHEL ライブカーネル、initramfs、および rootfs、および Ignition ファイルを HTTP または HTTPS サーバーに移動します。



注記

Ignition ファイルは OpenShift Container Platform インストーラーによって生成されます。

3. RHEL カーネル、initramfs、および Ignition ファイル、新規ディスクイメージ、および調整された parm 引数を使用して、新規 KVM ゲストノードを作成します。
   - **--location** には、HTTP サーバーまたは HTTPS サーバーのカーネル/initrd の場所を指定します。

- **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

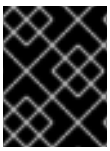
```
$ virt-install \
  --connect qemu:///system \
  --name {vm_name} \
  --vcpus {vcpus} \
  --memory {memory_mb} \
  --disk {vm_name}.qcow2,size={image_size| default(10,true)} \
  --network network={virt_network_parm} \
  --boot hd \
  --location {media_location},kernel={rhcos_kernel},initrd={rhcos_initrd} \
  --extra-args "rd.neednet=1 coreos.inst.install_dev=/dev/vda coreos.live.rootfs_url=
{rhcos_liveos} ip={ip}::{default_gateway}:{subnet_mask_length}:{vm_name}:enc1:none:
{MTU} nameserver={dns} coreos.inst.ignition_url={rhcos_ign}" \
  --noautoconsole \
  --wait
```

18.4.12.5. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

18.4.12.5.1. ISO インストールのネットワークオプション

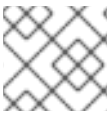
ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。



重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の表は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、 **ip=** および **nameserver=** カーネル引数の使用方法について説明します。



注記

カーネル引数 (**ip=** and **nameserver=**) を追加するときは、順序付けが重要です。

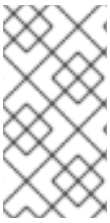
ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

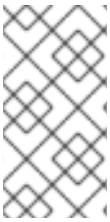
- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1つまたは複数のネットワークを設定する場合、1つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

18.4.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

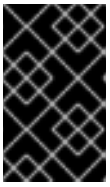
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

18.4.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

18.4.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
```

```
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

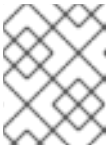
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

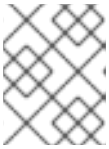
6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
```

```
master-2  Ready    master  74m  v1.29.4
worker-0  Ready    worker  11m  v1.29.4
worker-1  Ready    worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

18.4.16. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
```

```
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

18.4.16.1. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

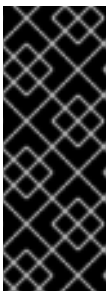
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

18.4.16.1.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z® にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

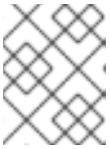
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION           AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16              True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

18.4.16.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

18.4.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

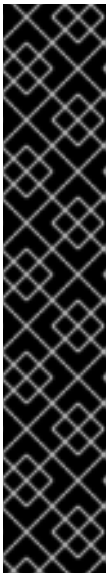
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

18.4.18. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。
- How to generate SOSREPORT within OpenShift4 nodes without SSH

18.4.19. 次のステップ

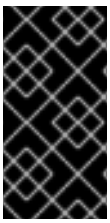
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

18.5. ネットワークが制限された環境での RHEL KVM のあるクラスターの IBM Z および IBM LINUXONE へのインストール

OpenShift Container Platform バージョン 4.16 では、制限されたネットワーク内で、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャーに、クラスターをインストールできます。

注記

このドキュメントは IBM Z® のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。



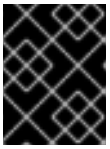
重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

18.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。



重要

インストールメディアにアクセスできるマシンからインストール手順が実行されるようにします。

- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

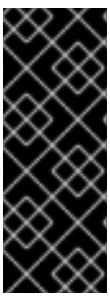
- 論理パーティション (LPAR) でホストされ、RHEL 8.6 以降をベースとする RHEL Kernel Virtual Machine (KVM) システムをプロビジョニングしている。Red Hat Enterprise Linux 8 and 9 Life Cycle を参照してください。

18.5.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

18.5.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

18.5.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

18.5.4. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

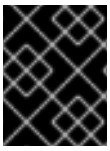
RHEL 8.6 以降をベースとする 1 つ以上の KVM ホストマシン。各 RHEL KVM ホストマシンで libvirt がインストールされ、実行している必要があります。仮想マシンは、各 RHEL KVM ホストマシンでプロビジョニングされます。

18.5.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表18.58 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる RHEL インスタンスにコントロールプレーンマシンを分散します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

18.5.4.2. ネットワーク接続の要件

OpenShift Container Platform インストーラーは、すべての Red Hat Enterprise Linux CoreOS (RHCOS) 仮想マシンに必要な Ignition ファイルを作成します。OpenShift Container Platform の自動インストールはブートストラップマシンで実行されます。これは各ノードで OpenShift Container Platform のインストールを開始し、Kubernetes クラスターを起動してから終了します。このブートストラップ時に、仮想マシンには Dynamic Host Configuration Protocol (DHCP) サーバーまたは静的 IP アドレスでネットワーク接続を確立している必要があります。

18.5.4.3. IBM Z ネットワーク接続の要件

RHEL KVM の IBM Z® にインストールするには、以下が必要です。

- OSA または RoCE ネットワークアダプターで設定された RHEL KVM ホスト。
- libvirt または MacVTap のブリッジネットワークを使用してネットワークをゲストに接続するように設定されているいずれかの RHEL KVM ホスト。
  仮想ネットワーク接続の種類 を参照してください。

18.5.4.4. ホストマシンのリソース要件

お使いの環境の RHEL KVM ホストは、OpenShift Container Platform 環境用に計画している仮想マシンをホストするために以下の要件を満たす必要があります。仮想化の使用開始 を参照してください。

OpenShift Container Platform バージョン 4.16 は、以下の IBM® ハードウェアにインストールできます。

- IBM® z16 (すべてのモデル)、IBM® z15 (すべてのモデル)、IBM® z14 (すべてのモデル)
- IBM® LinuxONE 4 (すべてのモデル)、IBM® LinuxONE III (すべてのモデル)、IBM® LinuxONE Empire II、IBM® LinuxONE Rockhopper II

18.5.4.5. 最小の IBM Z システム環境

ハードウェア要件

- クラスターごとに、SMT2 対応の 6 つの Integrated Facilities for Linux (IFL) に相当します。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z® の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。

重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR のウェイト管理、エンタイトルメント、および CPU 共有が重要な役割を果たします。

オペレーティングシステム要件

- libvirt で管理される、KVM を使用する RHEL 8.6 以降で実行する 1 つの LPAR。

RHEL KVM ホストで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

18.5.4.6. 最小リソース要件

それぞれのクラスターの仮想マシンは、以下の最小要件を満たしている必要があります。

仮想マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
Compute	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

18.5.4.7. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

オペレーティングシステム要件

- 高可用性が必要な場合は、libvirt で管理される、KVM を使用する RHEL 8.6 以降で実行する 2 または 3 つの LPAR。

RHEL KVM ホストで、以下を設定します。

- OpenShift Container Platform コンピュートプレーンマシン用に 3 つのゲスト仮想マシン (RHEL KVM ホストマシン全体に分散)
- OpenShift Container Platform コンピュートマシン用に 6 つ以上のゲスト仮想マシン (RHEL KVM ホストマシン全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン
- オーバーコミット環境で必須コンポーネントの可用性を確保するには、**cpu_shares** を使用してコントロールプレーンの優先度を引き上げます。インフラストラクチャーノードが存在する場合は、同じ操作を行います。IBM® ドキュメントの schedinfo を参照してください。

18.5.4.8. 優先されるリソース要件

各クラスターの仮想マシンについての優先される要件は以下の通りです。

仮想マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	8	16 GB	120 GB
Compute	RHCOS	6	8 GB	120 GB

18.5.4.9. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- IBM Z® および IBM® LinuxONE 環境に推奨されるホストプラクティス

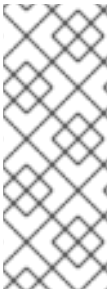
18.5.4.10. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起

動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。



注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

18.5.4.10.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

18.5.4.10.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表18.59 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック

プロトコル	ポート	説明
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表18.60 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表18.61 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

18.5.4.11. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表18.62 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

18.5.4.11.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例18.10 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例18.11 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

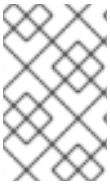
7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

18.5.4.12. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.63 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

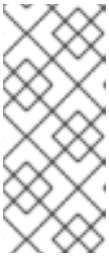
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.64 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



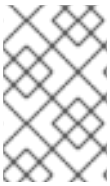
注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

18.5.4.12.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例18.12 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
```

```
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (5)
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (6)
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

(1) ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

(2)(4) ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

(3) ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

(5) ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

(6)

ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

18.5.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。
   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

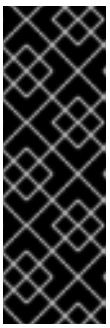
注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. Red Hat Enterprise Linux CoreOS (RHCOS) の 高速インストールまたは Red Hat Enterprise Linux CoreOS (RHCOS) のフルインストールのいずれかの実行を選択します。フルインストールでは、HTTP または HTTPS サーバーを設定し、Ignition ファイルを提供し、イメージをクラスターノードにインストールする必要があります。高速インストールの場合、HTTP または HTTPS サーバーは必要はありませんが、DHCP サーバーが必要です。高速インストール: Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成およびフルインストール: Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成のセクションを参照してください。

3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。



重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

5. クラスターに必要な DNS インフラストラクチャーを設定します。

a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

6. DNS 設定を検証します。

    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

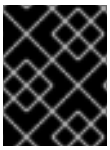
7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

18.5.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

    a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

    ```
    $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
    ```

    1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

    出力例

```
api.ocp4.example.com.  604800 IN A 192.168.1.5
```

b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

出力例

```
api-int.ocp4.example.com.  604800 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

18.5.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

–

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

18.5.8. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

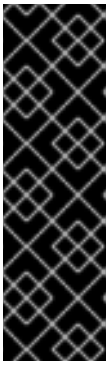
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Z® のインストール設定パラメーター

18.5.8.1. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

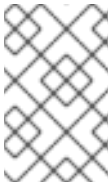
```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture: s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture: s390x
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
```

```
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
additionalTrustBundle: | (17)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (18)
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



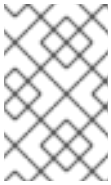
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が OpenShift Container Platform ノードで利用できない場合、**hyperthreading** パラメーターは影響を受けません。

重要

OpenShift Container Platform ノードまたは **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

(4) OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



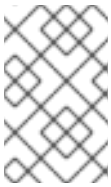
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Z® インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

18 リポジトリーのミラーリングに使用したコマンドの出力に従って、**imageContentSources** セクションを指定します。

重要

- **oc adm release mirror** コマンドを使用する場合は、**imageContentSources** セクションの出力を使用します。
- **oc mirror** コマンドを使用する場合は、コマンドの実行によって生成される **ImageContentSourcePolicy** ファイルの **repositoryDigestMirrors** セクションを使用します。
- **ImageContentSourcePolicy** は非推奨になりました。詳細は、**イメージレジストリーリポジトリーミラーリングの設定** を参照してください。

18.5.8.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

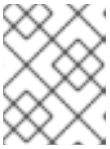
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

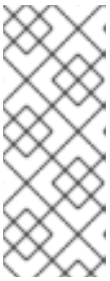
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** で

クラスターの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

18.5.8.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定される最小の 3 つのノードクラスターにゼロコンピューティングマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

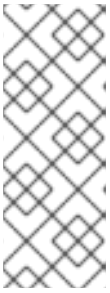
手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。



注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

18.5.9. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

18.5.9.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表18.65 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表18.66 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表18.67 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。

フィールド	型	説明
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表18.68 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表18.69 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表18.70 **policyAuditConfig** オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表18.71 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表18.72 gatewayConfig.ipv4 object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表18.73 gatewayConfig.ipv6 object

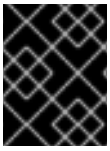
フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表18.74 ipsecConfig オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```



重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表18.75 kubeProxyConfig オブジェクト

フィールド	型	説明

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。 **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: ``` kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s ```

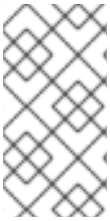
18.5.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.5.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform をプロビジョニングする IBM Z® インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を Red Hat Enterprise Linux (RHEL) ゲスト仮想マシンとしてインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

事前にパッケージ化された QEMU コピーオンライト (QCOW2) ディスクイメージを使用する RHCOS の高速インストールを実行できます。または、新規の QCOW2 ディスクイメージでフルインストールを実行できます。

システムのセキュリティーをさらに強化するために、ファストトラックインストールに進む前に、オプションで IBM® Secure Execution を使用して RHCOS をインストールできます。

18.5.11.1. IBM Secure Execution を使用した RHCOS のインストール

IBM® Secure Execution を使用して RHCOS をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

前提条件

- IBM® z15 以降、または IBM® LinuxONE III 以降。
- Red Hat Enterprise Linux (RHEL) 8 以降
- ブートストラップ Ignition ファイルがあります。ファイルは保護されていないため、他のユーザーが表示および編集できます。
- インストール後にブートイメージが変更されていないことを確認しました。
- すべてのノードを IBM® Secure Execution ゲストとして実行する必要があります。

手順

1. RHEL KVM ホストを準備して、IBM® Secure Execution をサポートします。

   - デフォルトでは、KVM ホストは IBM® Secure Execution モードのゲストをサポートしません。ゲストを IBM® Secure Execution モードでサポートするには、KVM ホストを LPAR モードで起動し、カーネルパラメーターを **prot_virt=1** に指定する必要があります。RHEL 8 で **prot_virt=1** を有効にするには、次の手順に従います。

     a. **/boot/loader/entries/** に移動して、ブートローダー設定ファイル ***.conf** を変更します。

     b. カーネルコマンドラインパラメーター **prot_virt=1** を追加します。

     c. **zipl** コマンドを実行し、システムを再起動します。
     IBM® Secure Execution for Linux のサポートを使用して正常に始動する KVM ホストは、次のカーネルメッセージを発行します。

     ```
     prot_virt: Reserving <amount>MB as ultravisor base storage.
     ```

     d. KVM ホストが IBM® Secure Execution をサポートするようになったことを確認するには、次のコマンドを実行します。

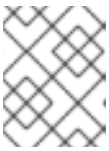
     ```
     # cat /sys/firmware/uv/prot_virt_host
     ```

     出力例

     ```
     1
     ```

     環境がセキュアホストの環境と一致していると検出された Linux インスタンスの場合、この属性の値は 1 です。他のインスタンスの場合、値は 0 です。

2. Ignition を介して KVM ゲストにホストキーを追加します。
最初の起動時に、RHCOS はホストキーを探して、それを使用して自身を再暗号化します。RHCOS は、**/etc/se-hostkeys** ディレクトリーで **ibm-z-hostkey-** で始まるファイルを検索します。クラスターが実行されている各マシンのすべてのホストキーは、管理者がディレクトリーにロードする必要があります。最初の起動後、他のマシンで VM を実行することはできません。

   注記

   安全なシステムで Ignition ファイルを準備する必要があります。例えば、別の IBM® Secure Execution ゲストです。

   以下に例を示します。

   ```
   {
     "ignition": { "version": "3.0.0" },
     "storage": {
       "files": [
         {
           "path": "/etc/se-hostkeys/ibm-z-hostkey-<your-hostkey>.crt",
           "contents": {
             "source": "data:;base64,<base64 encoded hostkey document>"
   ```

```
        },
        "mode": 420
      },
      {
        "path": "/etc/se-hostkeys/ibm-z-hostkey-<your-hostkey>.crt",
        "contents": {
          "source": "data:;base64,<base64 encoded hostkey document>"
        },
        "mode": 420
      }
    ]
  }
}
```

注記

ノードを複数の IBM Z® マシンで実行できるようにする場合は、必要な数のホストキーを追加できます。

3. Base64 でエンコードされた文字列を生成するには、次のコマンドを実行します。

```
base64 <your-hostkey>.crt
```

IBM® Secure Execution を実行していないゲストと比較すると、イグニションフェーズの前にイメージ全体がランダムに生成された LUKS パスフレーズで暗号化されるため、マシンの最初のブートに時間がかかります。

4. Ignition 保護の追加
Ignition 設定ファイルに保存されているシークレットが読み取られたり変更されたりしないようにするには、Ignition 設定ファイルを暗号化する必要があります。

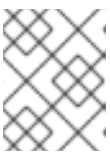
注記

必要なセキュリティーを実現するために、IBM® Secure Execution の実行時は、Ignition ロギングとローカルログインがデフォルトで無効になります。

a. 次のコマンドを実行して、**secex-qemu.qcow2** イメージの公開 GPG キーを取得し、そのキーを使用して Ignition 設定を暗号化します。

```
gpg --recipient-file /path/to/ignition.gpg.pub --yes --output /path/to/config.ign.gpg --
verbose --armor --encrypt /path/to/config.ign
```

5. RHCOS の高速インストールに従って、IBM® Secure Execution QCOW イメージを使用してノードをインストールします。



注記

仮想マシンを起動する前に、**serial=ignition** を **serial=ignition_crypted** に置き換え、**launchSecurity** パラメーターを追加します。

検証

RHCOS の高速インストールが完了し、最初の起動時に Ignition が実行したら、復号が成功したかどうかを確認します。

- 復号が成功すると、次の例のような出力が期待できます。

  出力例

  ```
  [    2.801433] systemd[1]: Starting coreos-ignition-setup-user.service - CoreOS Ignition User Config Setup...

  [    2.803959] coreos-secex-ignition-decrypt[731]: gpg: key <key_name>: public key "Secure Execution (secex) 38.20230323.dev.0" imported
  [    2.808874] coreos-secex-ignition-decrypt[740]: gpg: encrypted with rsa4096 key, ID <key_name>, created <yyyy-mm-dd>
  [  OK  ] Finished coreos-secex-igni…S Secex Ignition Config Decryptor.
  ```

- 復号nい失敗した場合は、次の例のような出力が予想されます。

  出力例

  ```
  Starting coreos-ignition-s…reOS Ignition User Config Setup...
  [    2.863675] coreos-secex-ignition-decrypt[729]: gpg: key <key_name>: public key "Secure Execution (secex) 38.20230323.dev.0" imported
  [    2.869178] coreos-secex-ignition-decrypt[738]: gpg: encrypted with RSA key, ID <key_name>
  [    2.870347] coreos-secex-ignition-decrypt[738]: gpg: public key decryption failed: No secret key
  [    2.870371] coreos-secex-ignition-decrypt[738]: gpg: decryption failed: No secret key
  ```

関連情報

- IBM® Secure Execution for Linux の紹介
- IBM® Secure Execution ホストまたはゲストとしての Linux
- IBM Z での IBM® Secure Execution の設定

18.5.11.2. IBM Z または IBM& LinuxONE 環境での静的 IP を使用した NBDE の設定

IBM Z® または IBM® LinuxONE 環境で NBDE ディスク暗号化を有効にするには、追加の手順が必要です。このセクションで詳しく説明します。

前提条件

- External Tang Server がセットアップされました。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。
- **butane** ユーティリティーをインストールしている。
- Butane でマシン設定を作成する手順を確認している。

手順

1. コントロールプレーンとコンピュートノードの Butane 設定ファイルを作成します。

次のコントロールプレーンノードの Butane 設定の例では、ディスク暗号化用に **master-storage.bu** という名前のファイルを作成します。

```
variant: openshift
version: 4.16.0
metadata:
  name: master-storage
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  luks:
    - clevis:
        tang:
          - thumbprint: QcPr_NHFJammnRCA3fFMVdNBwjs
            url: http://clevis.example.com:7500
        options: 1
          - --cipher
          - aes-cbc-essiv:sha256
      device: /dev/disk/by-partlabel/root
      label: luks-root
      name: root
      wipe_volume: true
  filesystems:
    - device: /dev/mapper/root
      format: xfs
      label: root
      wipe_filesystem: true
openshift:
  fips: true 2
```

1 暗号オプションは、FIPS モードが有効な場合にのみ必要です。FIPS が無効になっている場合は、エントリーを省略します。

2 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

2. 次のコマンドを実行して、マシンを起動するためのカスタマイズされた initramfs ファイルを作成します。

```
$ coreos-installer pxe customize \
    /root/rhcos-bootfiles/rhcos-<release>-live-initramfs.s390x.img \
    --dest-device /dev/disk/by-id/scsi-<serial_number> --dest-karg-append \
    ip=<ip_address>::<gateway_ip>:<subnet_mask>::<network_device>:none \
    --dest-karg-append nameserver=<nameserver_ip> \
    --dest-karg-append rd.neednet=1 -o \
    /root/rhcos-bootfiles/<node_name>-initramfs.s390x.img
```

注記

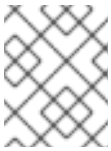
最初のブートの前に、クラスター内の各ノードの initramfs をカスタマイズし、PXE カーネルパラメーターを追加する必要があります。

3. **ignition.platform.id=metal** および **ignition.firstboot** を含むパラメーターファイルを作成します。

```
rd.neednet=1 \
console=ttysclp0 \
ignition.firstboot ignition.platform.id=metal \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \ 1
coreos.inst.ignition_url=http://<http_server>/master.ign \ 2
ip=10.19.17.2::10.19.17.1:255.255.255.0::enbdd0:none nameserver=10.19.17.1 \
zfcp.allow_lun_scan=0 \
rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
rd.zfcp=0.0.5677,0x600606680g7f0056,0x034F000000000000
```

1 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

2 Ignition 設定ファイルの場所を指定します。**master.ign** または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。



注記

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

関連情報

- Butane でのマシン設定の作成

18.5.11.3. 事前にパッケージ化された QCOW2 ディスクイメージを使用した高速インストール

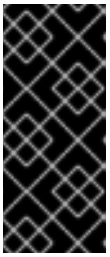
Red Hat Enterprise Linux CoreOS (RHCOS) の高速インストールでマシンを作成するには、事前にパッケージ化された Red Hat Enterprise Linux CoreOS (RHCOS)QEMU コピーオンライト (QCOW2) ディスクイメージをインポートします。

前提条件

- この手順では RHEL KVM ホストと呼ばれる、KVM を使用する RHEL 8.6 以降で実行されている少なくとも 1 つの LPAR。
- KVM/QEMU ハイパーバイザーが RHEL KVM ホストにインストーされている
- ノードのホスト名および逆引き参照を実行できるドメインネームサーバー (DNS)。
- IP アドレスを提供する DHCP サーバー。

手順

1. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラーページから RHEL QEMU コピーオンライト (QCOW2) ディスクイメージファイルを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な RHCOS QCOW2 イメージのみを使用します。

2. QCOW2 ディスクイメージおよび Ignition ファイルを RHEL KVM ホストの共通ディレクトリーにダウンロードします。
例: **/var/lib/libvirt/images**

注記

Ignition ファイルは OpenShift Container Platform インストーラーによって生成されます。

3. 各 KVM ゲストノードの QCOW2 ディスクイメージバッキングファイルで、新しいディスクイメージを作成します。

```
$ qemu-img create -f qcow2 -F qcow2 -b /var/lib/libvirt/images/{source_rhcos_qemu}
/var/lib/libvirt/images/{vmname}.qcow2 {size}
```

4. Ignition ファイルと新規ディスクイメージを使用して、新規 KVM ゲストノードを作成します。

```
$ virt-install --noautoconsole \
  --connect qemu:///system \
  --name {vm_name} \
  --memory {memory} \
  --vcpus {vcpus} \
  --disk {disk} \
  --launchSecurity type="s390-pv" \ 1
  --import \
  --network network={network},mac={mac} \
  --disk path={ign_file},format=raw,readonly=on,serial=ignition,startup_policy=optional 2
```

1. IBM® Secure Execution が有効な場合は、**launchSecurity type="s390-pv"** パラメーターを追加します。

2. IBM® Secure Execution が有効な場合は、**serial=ignition** を **Serial=ignition_crypted** に置き換えます。

18.5.11.4. 新規 QCOW2 ディスクイメージへのフルインストール

新規 QEMU copy-on-write (QCOW2) ディスクイメージのフルインストールでマシンを作成するには、以下の手順を実施します。

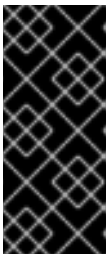
前提条件

- この手順では RHEL KVM ホストと呼ばれる、KVM を使用する RHEL 8.6 以降で実行されている少なくとも 1 つの LPAR。
- KVM/QEMU ハイパーバイザーが RHEL KVM ホストにインストーされている

- ノードのホスト名および逆引き参照を実行できるドメインネームサーバー (DNS)。
- HTTP または HTTPS サーバーが設定されている。

手順

1. Red Hat カスタマーポータルの 製品のダウンロード ページ、または RHCOS イメージミラー ページから RHEL カーネル、initramfs、および rootfs ファイルを取得します。



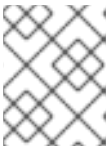
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な RHCOS QCOW2 イメージのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

2. **virt-install** を起動する前に、ダウンロードした RHEL ライブカーネル、initramfs、および rootfs、および Ignition ファイルを HTTP または HTTPS サーバーに移動します。



注記

Ignition ファイルは OpenShift Container Platform インストーラーによって生成されます。

3. RHEL カーネル、initramfs、および Ignition ファイル、新規ディスクイメージ、および調整された parm 引数を使用して、新規 KVM ゲストノードを作成します。

- **--location** には、HTTP サーバーまたは HTTPS サーバーのカーネル/initrd の場所を指定します。
- **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
- **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

```
$ virt-install \
  --connect qemu:///system \
  --name {vm_name} \
  --vcpus {vcpus} \
  --memory {memory_mb} \
  --disk {vm_name}.qcow2,size={image_size| default(10,true)} \
  --network network={virt_network_parm} \
```

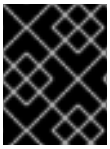
```
   --boot hd \
   --location {media_location},kernel={rhcos_kernel},initrd={rhcos_initrd} \
   --extra-args "rd.neednet=1 coreos.inst.install_dev=/dev/vda coreos.live.rootfs_url=
{rhcos_liveos} ip={ip}::{default_gateway}:{subnet_mask_length}:{vm_name}:enc1:none:
{MTU} nameserver={dns} coreos.inst.ignition_url={rhcos_ign}" \
   --noautoconsole \
   --wait
```

18.5.11.5. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

18.5.11.5.1. ISO インストールのネットワークオプション

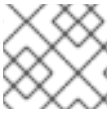
ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。



重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の表は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、 **ip=** および **nameserver=** カーネル引数の使用方法について説明します。



注記

カーネル引数 (**ip=** and **nameserver=**) を追加するときは、順序付けが重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

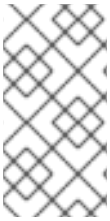
次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**

- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

  ```
  ip=::10.10.10.254::::
  ```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

  ```
  ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
  vlan=enp2s0.100:enp2s0
  ```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

  ```
  ip=enp2s0.100:dhcp
  vlan=enp2s0.100:enp2s0
  ```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

18.5.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。

- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

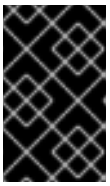
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

18.5.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

18.5.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
master-2 Ready    master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

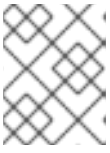
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

18.5.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

18.5.15.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

18.5.15.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

18.5.15.2.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z® にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

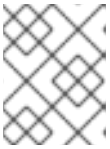
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME           VERSION        AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16             True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

18.5.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

18.5.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

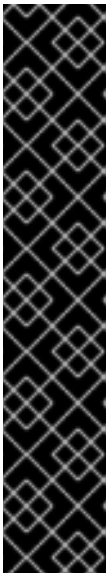
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                               1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                               1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
Running    0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

4. Cluster registration ページでクラスターを登録します。

関連情報

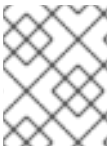
- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH

18.5.17. 次のステップ

- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。

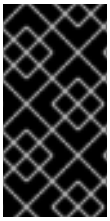
18.6. IBM Z および IBM LINUXONE 上の LPAR へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャー上の論理パーティション (LPAR) に、クラスターをインストールできます。



注記

このドキュメントは IBM Z® のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

18.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

18.6.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

18.6.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

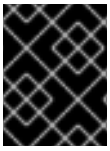
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

18.6.3.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表18.76 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

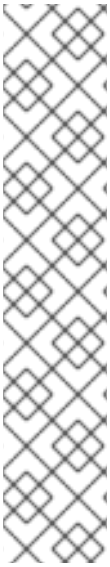
18.6.3.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表18.77 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1 秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
Compute	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

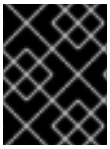
関連情報

- ストレージの最適化

18.6.3.3. 最小の IBM Z システム環境

OpenShift Container Platform バージョン 4.16 は、以下の IBM® ハードウェアにインストールできます。

- IBM® z16 (すべてのモデル)、IBM® z15 (すべてのモデル)、IBM® z14 (すべてのモデル)
- IBM® LinuxONE 4 (すべてのモデル)、IBM® LinuxONE III (すべてのモデル)、IBM® LinuxONE Empire II、IBM® LinuxONE Rockhopper II



重要

ハイパーバイザーなしで IBM Z® 上で OpenShift Container Platform を実行する場合は、Dynamic Partition Manager (DPM) を使用してマシンを管理します。

ハードウェア要件

- クラスターごとに、SMT2 対応の 6 つの Integrated Facilities for Linux (IFL) に相当します。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z® の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。

重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR のウェイト管理、エンタイトルメント、および CPU 共有が重要な役割を果たします。

オペレーティングシステム要件

- 5 つの論理パーティション (LPAR)
  - OpenShift Container Platform コントロールプレーンマシン用の 3 つの LPAR
  - OpenShift Container Platform コンピュートマシン用の 2 つの LPAR
- 一時的な OpenShift Container Platform ブートストラップマシン用の 1 台のマシン

IBM Z ネットワーク接続の要件
IBM Z® の LPAR 内にインストールするには、次のものが必要です。

- 直接接続された OSA または RoCE ネットワークアダプター
- 推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

ディスクストレージ

- FICON 接続のディスクストレージ (DASD)。これには専用の DASD を使用できます。その場合、デフォルトの CDL 形式でフォーマットされている必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

関連情報

- PR/SM モードの考慮事項については、IBM® Documentation の Processors Resource/Systems Manager Planning Guide を参照してください。
- DPM モードの考慮事項については、IBM® Documentation の IBM Dynamic Partition Manager (DPM) Guide を参照してください。
- LPAR のウェイト管理とエンタイトルメントについては、Topics in LPAR Performance を参照してください。
- IBM Z® および IBM® LinuxONE 環境に推奨されるホストプラクティス

18.6.3.4. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。
- HiperSocket。デバイスとしてノードに直接割り当てたもの。HiperSockets をノードに直接接続するには、RHEL 8 ゲストを介して外部ネットワークへのゲートウェイをセットアップし、HiperSockets ネットワークにブリッジする必要があります。

オペレーティングシステム要件

- OpenShift Container Platform コントロールプレーンマシン用の 3 つの LPAR
- OpenShift Container Platform コンピュートマシン用の 6 つ以上の LPAR
- 一時的な OpenShift Container Platform ブートストラップマシン用の 1 つのマシンまたは LPAR

IBM Z ネットワーク接続の要件
IBM Z® の LPAR 内にインストールするには、次のものが必要です。

- 直接接続された OSA または RoCE ネットワークアダプター
- 推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

ディスクストレージ

- FICON 接続のディスクストレージ (DASD)。これには専用の DASD を使用できます。その場合、デフォルトの CDL 形式でフォーマットされている必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

18.6.3.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

18.6.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

18.6.3.6.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表18.78 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット

プロトコル	ポート	説明
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表18.79 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表18.80 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

18.6.3.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表18.81 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>. <base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>. <base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>. <base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps. <cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。

コンポーネント	レコード	説明
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

18.6.3.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例18.13 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
```

```
  1W ) ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

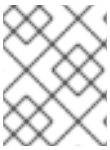
ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例18.14 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1. Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。
2. Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。
3. ブートストラップマシンの逆引き DNS 解決を提供します。
4. 5. 6. コントロールプレーンマシンの逆引き DNS 解決を提供します。
7. 8. コンピュートマシンの逆引き DNS 解決を提供します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

18.6.3.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーイン

フラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.82 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

    - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
    - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.83 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

注記

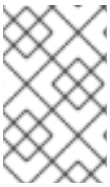
ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

18.6.3.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューショ

ンを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例18.15 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2 rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
```

```
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 5
    bind *:443
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:443 check inter 1s
    server compute1 compute1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 6
    bind *:80
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:80 check inter 1s
    server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

18.6.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用

の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

5. クラスターに必要な DNS インフラストラクチャーを設定します。
   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

18.6.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

–

出力例

```
api-int.ocp4.example.com.  604800 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

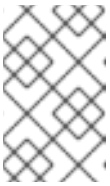
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

18.6.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

18.6.7. インストールプログラムの取得

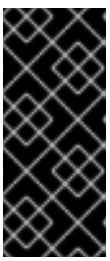
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

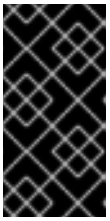
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

18.6.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

18.6.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

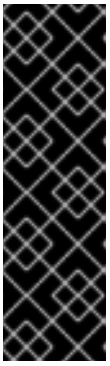
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Z® のインストール設定パラメーター

18.6.9.1. IBM Z のサンプル install-config.yaml ファイル

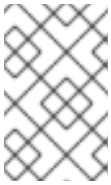
install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture: s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture: s390x
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths": ...}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

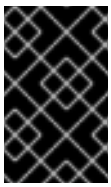
(2) (5) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



注記

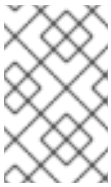
同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が OpenShift Container Platform ノードで利用できない場合、**hyperthreading** パラメーターは影響を受けません。



重要

OpenShift Container Platform ノードまたは **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



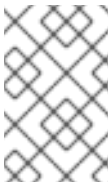
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Z® インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

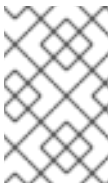
重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

18.6.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーを

バイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップ

を参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

18.6.9.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定される最小の 3 つのノードクラスターにゼロコンピューティングマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

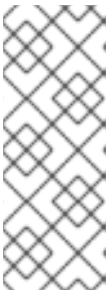
手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。



注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

18.6.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

18.6.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表18.84 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表18.85 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表18.86 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。

フィールド	型	説明
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表18.87 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表18.88 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表18.89 **policyAuditConfig** オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表18.90 **gatewayConfig** オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表18.91 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表18.92 **gatewayConfig.ipv6** object

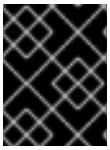
フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表18.93 **ipsecConfig** オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```



重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)

kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表18.94 **kubeProxyConfig** オブジェクト

フィールド	型	説明

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。 **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: ``` kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s ```

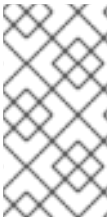
18.6.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

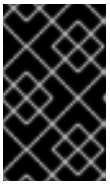
1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.6.12. IBM Z または IBM& LinuxONE 環境での静的 IP を使用した NBDE の設定

IBM Z® または IBM® LinuxONE 環境で NBDE ディスク暗号化を有効にするには、追加の手順が必要です。このセクションで詳しく説明します。

前提条件

- External Tang Server がセットアップされました。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。
- **butane** ユーティリティーをインストールしている。
- Butane でマシン設定を作成する手順を確認している。

手順

1. コントロールプレーンとコンピュートノードの Butane 設定ファイルを作成します。
次のコントロールプレーンノードの Butane 設定の例では、ディスク暗号化用に **master-storage.bu** という名前のファイルを作成します。

```
variant: openshift
version: 4.16.0
metadata:
  name: master-storage
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  luks:
    - clevis:
        tang:
          - thumbprint: QcPr_NHFJammnRCA3fFMVdNBwjs
            url: http://clevis.example.com:7500
        options: 1
          - --cipher
          - aes-cbc-essiv:sha256
      device: /dev/disk/by-partlabel/root 2
```

```
      label: luks-root
      name: root
      wipe_volume: true
  filesystems:
    - device: /dev/mapper/root
      format: xfs
      label: root
      wipe_filesystem: true
openshift:
  fips: true (3)
```

(1) 暗号オプションは、FIPS モードが有効な場合にのみ必要です。FIPS が無効になっている場合は、エントリーを省略します。

(2) DASD タイプのディスクにインストールする場合は、**device:/dev/disk/by-label/root** に置き換えます。

(3) FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

2. 次のコマンドを実行して、マシンを起動するためのカスタマイズされた initramfs ファイルを作成します。

```
$ coreos-installer pxe customize \
    /root/rhcos-bootfiles/rhcos-<release>-live-initramfs.s390x.img \
    --dest-device /dev/disk/by-id/scsi-<serial_number> --dest-karg-append \
    ip=<ip_address>::<gateway_ip>:<subnet_mask>::<network_device>:none \
    --dest-karg-append nameserver=<nameserver_ip> \
    --dest-karg-append rd.neednet=1 -o \
    /root/rhcos-bootfiles/<node_name>-initramfs.s390x.img
```

注記

最初のブートの前に、クラスター内の各ノードの initramfs をカスタマイズし、PXE カーネルパラメーターを追加する必要があります。

3. **ignition.platform.id=metal** および **ignition.firstboot** を含むパラメーターファイルを作成します。

コントロールプレーンマシンのカーネルパラメーターファイルの例:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/dasda \(1)
ignition.firstboot ignition.platform.id=metal \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \(2)
coreos.inst.ignition_url=http://<http_server>/master.ign \(3)
ip=10.19.17.2::10.19.17.1:255.255.255.0::enbdd0:none nameserver=10.19.17.1 \
zfcp.allow_lun_scan=0 \(4)
rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
rd.zfcp=0.0.5677,0x600606680g7f0056,0x034F000000000000 \(5)
```

1. DASD タイプのディスクにインストールする場合は、**coreos.inst.install_dev=/dev/dasda** を追加します。FCP タイプのディスクの場合は、この値を省略します。
2. 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
3. Ignition 設定ファイルの場所を指定します。**master.ign** または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
4. FCP タイプのディスクにインストールする場合は、**zfcp.allow_lun_scan=0** を追加します。DASD タイプのディスクの場合は、この値を省略します。
5. DASD タイプのディスクにインストールする場合は、**rd.dasd=0.0.3490** に置き換えて DASD デバイスを指定します。

注記

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

関連情報

- Butane でのマシン設定の作成

18.6.13. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

独自にプロビジョニングする IBM Z® インフラストラクチャーに OpenShift Container Platform をインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を LPAR にインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、RHCOS ゲストマシンの再起動後に、OpenShift Container Platform ブートストラッププロセスが自動的に開始します。

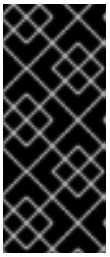
マシンを作成するには、以下の手順を実行します。

前提条件

- 作成するマシンがアクセスできるプロビジョニングマシンで稼働している HTTP または HTTPS サーバー。
- セキュアブートを有効にする場合は、適切な Product Signing Key を取得し、IBM ドキュメントの Secure boot on IBM Z and IBM LinuxONE を確認した。

手順

1. プロビジョニングマシンで Linux にログインします。
2. RHCOS イメージミラー から Red Hat Enterprise Linux CoreOS (RHCOS) カーネル、 initramfs および rootfs ファイルを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な kernel、initramfs、および rootfs アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

注記

rootfs イメージは FCP および DASD の場合と同じです。

3. パラメーターファイルを作成します。以下のパラメーターは特定の仮想マシンに固有のものです。
   - **ip=** には、以下の 7 つのエントリーを指定します。
     i. マシンの IP アドレス。
     ii. 空の文字列。
     iii. ゲートウェイ。
     iv. ネットマスク。
     v. **hostname.domainname** 形式のマシンホストおよびドメイン名。この値を省略して、RHCOS に決定させるようにします。
     vi. ネットワークインターフェイス名。この値を省略して、RHCOS に決定させるようにします。
     vii. 静的 IP アドレスを使用する場合、 **none** を指定します。
   - **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
   - **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
   - オプション: セキュアブートを有効にするには、**coreos.inst.secure_ipl** を追加します。
   - DASD タイプのディスクへのインストールには、以下のタスクを実行します。
     i. **coreos.inst.install_dev=** には、**/dev/dasda** を指定します。
     ii. **rd.dasd=** を使用して、RHCOS がインストールされる DASD を指定します。

iii. その他のパラメーターはすべて変更しません。
ブートストラップマシンのパラメーターファイルのサンプル **bootstrap-0.parm**:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/dasda \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.
<architecture>.img \(1)
coreos.inst.ignition_url=http://<http_server>/bootstrap.ign \(2)
coreos.inst.secure_ipl \(3)
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.dasd=0.0.3490
```

(1) 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

(2) Ignition 設定ファイルの場所を指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

(3) オプション: セキュアブートを有効にするには、**coreos.inst.secure_ipl** を追加します。

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

- FCP タイプのディスクへのインストールには、以下のタスクを実行します。

i. **rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して RHCOS がインストールされる FCP ディスクを指定します。マルチパスの場合、それぞれの追加のステップについてこのステップを繰り返します。

注記

複数のパスを使用してインストールする場合は、問題が発生する可能性があるため、後でではなくインストールの直後にマルチパスを有効にする必要があります。

ii. インストールデバイスを **coreos.inst.install_dev=/dev/disk/by-id/scsi-<serial_number>** として設定します。

注記

追加の LUN が NPIV で設定される場合は、FCP に **zfcp.allow_lun_scan=0** が必要です。CSI ドライバーを使用するために **zfcp.allow_lun_scan=1** を有効にする必要がある場合などには、各ノードが別のノードのブートパーティションにアクセスできないように NPIV を設定する必要があります。

iii. その他のパラメーターはすべて変更しません。

重要

マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

以下は、マルチパスが設定されたコンピュートノードのパラメーターファイルのサンプル **worker-1.parm** です。

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/disk/by-id/scsi-<serial_number> \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.
<architecture>.img \
coreos.inst.ignition_url=http://<http_server>/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

4. FTP などを使用し、initramfs、kernel、パラメーターファイル、および RHCOS イメージを LPAR に転送します。FTP でファイルを転送して起動する方法については、LPAR へのインストール を参照してください。

5. マシンを起動します。

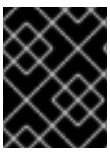
6. クラスター内の他のマシンについてこの手順を繰り返します。

18.6.13.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

18.6.13.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

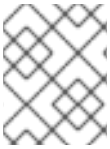


重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS イメージでネットワークおよびボンディングを設定する例

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**

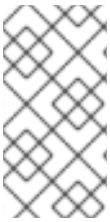
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、**options** はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードを使用する場合の問題を回避するために、常にアクティブバックアップモードで **fail_over_mac=1** オプションを設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1、em2**) のコンマ区切りリストを表します。

注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

18.6.14. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

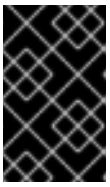
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

18.6.15. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

18.6.16. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.29.4
   master-1  Ready     master  73m  v1.29.4
   master-2  Ready     master  74m  v1.29.4
   worker-0  Ready     worker  11m  v1.29.4
   worker-1  Ready     worker  11m  v1.29.4
   ```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

18.6.17. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

18.6.17.1. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

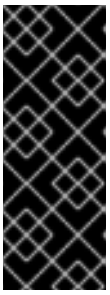
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

18.6.17.1.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z® にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION        AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16              True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

18.6.17.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

18.6.18. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
```

```
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver             apiserver-z25h4                             1/1     Running    0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
Running    0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace>
```
(1)

(1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

検証

OpenShift Container Platform のブートストラッププロセス中にセキュアブートを有効にした場合は、次の検証手順が必要です。

1. 次のコマンドを実行してノードをデバッグします。

```
$ oc debug node/<node_name>
chroot /host
```

2. 次のコマンドを実行して、セキュアブートが有効になっていることを確認します。

```
$ cat /sys/firmware/ipl/secure
```

出力例

```
1
```
(1)

(1) セキュアブートが有効になっている場合、値は **1** です。有効になっていない場合は **0** です。

3. 次のコマンドを実行して、再 IPL 設定をリスト表示します。

```
# lsreipl
```

FCP ディスクの出力例

```
Re-IPL type: fcp
WWPN: 0x500507630400d1e3
LUN: 0x4001400e00000000
Device: 0.0.810e
bootprog: 0
br_lba: 0
Loadparm: ""
Bootparms: ""
clear: 0
```

DASD ディスクの出力例

```
for DASD output:
Re-IPL type: ccw
Device: 0.0.525d
Loadparm: ""
clear: 0
```

4. 次のコマンドを実行してノードをシャットダウンします。

```
sudo shutdown -h
```

5. Hardware Management Console (HMC) から LPAR からのブートを開始します。IBM ドキュメントの Initiating a secure boot from an LPAR を参照してください。

6. ノードが復帰したら、セキュアブートのステータスを再度確認します。

18.6.19. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。
- How to generate SOSREPORT within OpenShift4 nodes without SSH

18.6.20. 次のステップ

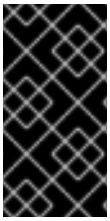
- RHCOS でカーネル引数を使用してマルチパスを有効化 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

18.7. ネットワークが制限された環境での IBM Z および IBM LINUXONE 上の LPAR へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、制限されたネットワーク内で、独自にプロビジョニングする IBM Z® または IBM® LinuxONE インフラストラクチャー上の論理パーティション (LPAR) に、クラスターをインストールできます。

注記

このドキュメントは IBM Z® のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。

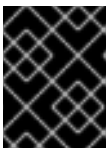


重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

18.7.1. 前提条件

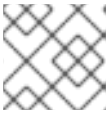
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ネットワークが制限された環境でインストールのミラーレジストリーを作成 し、お使いの OpenShift Container Platform のバージョンの **imageContentSources** データを取得している。
- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。



重要

インストールメディアにアクセスできるマシンからインストール手順が実行されるようにします。

- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。



注記

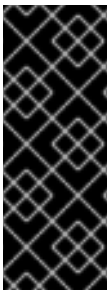
プロキシーを設定する場合は、このサイトリストも確認してください。

18.7.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

18.7.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

18.7.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。

- クラスターの更新を実行するために必要なパッケージを取得します。

18.7.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

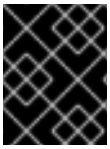
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

18.7.4.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表18.95 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

18.7.4.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表18.96 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS)
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
Compute	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

18.7.4.3. 最小の IBM Z システム環境

OpenShift Container Platform バージョン 4.16 は、以下の IBM® ハードウェアにインストールできます。

- IBM® z16 (すべてのモデル)、IBM® z15 (すべてのモデル)、IBM® z14 (すべてのモデル)
- IBM® LinuxONE 4 (すべてのモデル)、IBM® LinuxONE III (すべてのモデル)、IBM® LinuxONE Empire II、IBM® LinuxONE Rockhopper II

重要

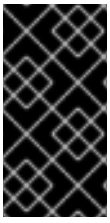
ハイパーバイザーなしで IBM Z® 上で OpenShift Container Platform を実行する場合は、Dynamic Partition Manager (DPM) を使用してマシンを管理します。

ハードウェア要件

- クラスターごとに、SMT2 対応の 6 つの Integrated Facilities for Linux (IFL) に相当します。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z® の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。



重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR のウェイト管理、エンタイトルメント、および CPU 共有が重要な役割を果たします。

オペレーティングシステム要件

- 5 つの論理パーティション (LPAR)
  - OpenShift Container Platform コントロールプレーンマシン用の 3 つの LPAR
  - OpenShift Container Platform コンピュートマシン用の 2 つの LPAR
- 一時的な OpenShift Container Platform ブートストラップマシン用の 1 台のマシン

IBM Z ネットワーク接続の要件
IBM Z® の LPAR 内にインストールするには、次のものが必要です。

- 直接接続された OSA または RoCE ネットワークアダプター
- 推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

ディスクストレージ

- FICON 接続のディスクストレージ (DASD)。これには専用の DASD を使用できます。その場合、デフォルトの CDL 形式でフォーマットされている必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

関連情報

- PR/SM モードの考慮事項については、IBM® Documentation の Processors Resource/Systems Manager Planning Guide を参照してください。
- DPM モードの考慮事項については、IBM® Documentation の IBM Dynamic Partition Manager (DPM) Guide を参照してください。
- LPAR のウェイト管理とエンタイトルメントについては、Topics in LPAR Performance を参照してください。
- IBM Z® および IBM® LinuxONE 環境に推奨されるホストプラクティス

18.7.4.4. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。
- HiperSocket。デバイスとしてノードに直接割り当てたもの。HiperSockets をノードに直接接続するには、RHEL 8 ゲストを介して外部ネットワークへのゲートウェイをセットアップし、HiperSockets ネットワークにブリッジする必要があります。

オペレーティングシステム要件

- OpenShift Container Platform コントロールプレーンマシン用の 3 つの LPAR
- OpenShift Container Platform コンピュートマシン用の 6 つ以上の LPAR
- 一時的な OpenShift Container Platform ブートストラップマシン用の 1 つのマシンまたは LPAR

IBM Z ネットワーク接続の要件
IBM Z® の LPAR 内にインストールするには、次のものが必要です。

- 直接接続された OSA または RoCE ネットワークアダプター
- 推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

ディスクストレージ

- FICON 接続のディスクストレージ (DASD)。これには専用の DASD を使用できます。その場合、デフォルトの CDL 形式でフォーマットされている必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

18.7.4.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

18.7.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

18.7.4.6.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

18.7.4.6.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表18.97 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表18.98 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表18.99 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

18.7.4.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

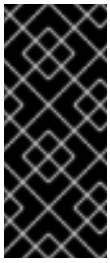
また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表18.100 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

18.7.4.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例18.16 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
```

```
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例18.17 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
```

```
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

18.7.4.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

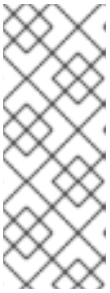
重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.101 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

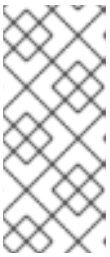
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表18.102 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

18.7.4.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例18.18 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
```

```
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 1
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup 2
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 3
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 4
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 5
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 6
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

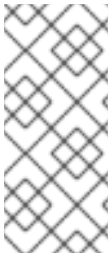
1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

18.7.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。

3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

5. クラスターに必要な DNS インフラストラクチャーを設定します。

   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

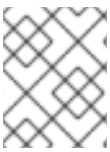
   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

6. DNS 設定を検証します。

   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

18.7.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

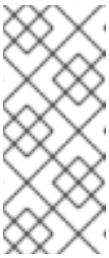
   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```



注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

18.7.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

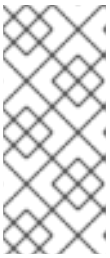
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

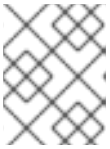
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

18.7.8. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

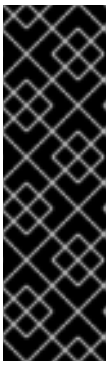
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

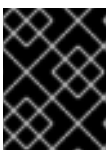
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Z® のインストール設定パラメーター

18.7.8.1. IBM Z のサンプル install-config.yaml ファイル

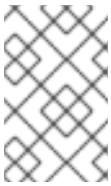
install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture: s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture: s390x
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
additionalTrustBundle: | (17)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (18)
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2)(5) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(3)(6) 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定しま



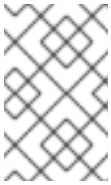
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が OpenShift Container Platform ノードで利用できない場合、**hyperthreading** パラメーターは影響を受けません。

重要

OpenShift Container Platform ノードまたは **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



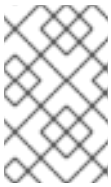
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Z® インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

18 リポジトリーのミラーリングに使用したコマンドの出力に従って、**imageContentSources** セクションを指定します。

重要

- **oc adm release mirror** コマンドを使用する場合は、**imageContentSources** セクションの出力を使用します。
- **oc mirror** コマンドを使用する場合は、コマンドの実行によって生成される **ImageContentSourcePolicy** ファイルの **repositoryDigestMirrors** セクションを使用します。
- **ImageContentSourcePolicy** は非推奨になりました。詳細は、**イメージレジストリーリポジトリーミラーリングの設定** を参照してください。

18.7.8.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

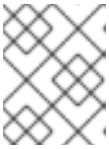
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
```

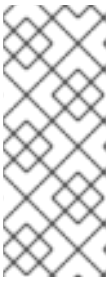
```
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

18.7.8.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定されるベアメタルクラスターに、ゼロコンピュートマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

18.7.9. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

18.7.9.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表18.103 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 `spec:` `  clusterNetwork:` `  - cidr: 10.128.0.0/19` `  hostPrefix: 23` `  - cidr: 10.128.32.0/19` `  hostPrefix: 23`
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 `spec:` `  serviceNetwork:` `  - 172.30.0.0/14` マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	型	説明
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表18.104 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表18.105 ovnKubernetesConfig オブジェクト

フィールド	型	説明

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表18.106 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表18.107 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表18.108 **policyAuditConfig** オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表18.109 **gatewayConfig** オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

フィールド	型	説明
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表18.110 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表18.111 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表18.112 **ipsecConfig** オブジェクト

フィールド	型	説明

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

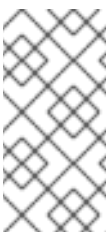
```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表18.113 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	型	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

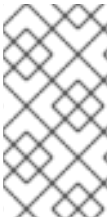
18.7.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
```

```
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.7.11. IBM Z または IBM& LinuxONE 環境での静的 IP を使用した NBDE の設定

IBM Z® または IBM® LinuxONE 環境で NBDE ディスク暗号化を有効にするには、追加の手順が必要です。このセクションで詳しく説明します。

前提条件

- External Tang Server がセットアップされました。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。
- **butane** ユーティリティーをインストールしている。
- Butane でマシン設定を作成する手順を確認している。

手順

1. コントロールプレーンとコンピュートノードの Butane 設定ファイルを作成します。次のコントロールプレーンノードの Butane 設定の例では、ディスク暗号化用に **master-storage.bu** という名前のファイルを作成します。

```
variant: openshift
version: 4.16.0
metadata:
  name: master-storage
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  luks:
    - clevis:
        tang:
          - thumbprint: QcPr_NHFJammnRCA3fFMVdNBwjs
            url: http://clevis.example.com:7500
        options: 1
          - --cipher
          - aes-cbc-essiv:sha256
      device: /dev/disk/by-partlabel/root 2
      label: luks-root
      name: root
      wipe_volume: true
  filesystems:
    - device: /dev/mapper/root
      format: xfs
      label: root
      wipe_filesystem: true
openshift:
  fips: true 3
```

1. 暗号オプションは、FIPS モードが有効な場合にのみ必要です。FIPS が無効になっている場合は、エントリーを省略します。
2. DASD タイプのディスクにインストールする場合は、**device:/dev/disk/by-label/root** に置き換えます。
3. FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

2. 次のコマンドを実行して、マシンを起動するためのカスタマイズされた initramfs ファイルを作成します。

```
$ coreos-installer pxe customize \
    /root/rhcos-bootfiles/rhcos-<release>-live-initramfs.s390x.img \
    --dest-device /dev/disk/by-id/scsi-<serial_number> --dest-karg-append \
    ip=<ip_address>::<gateway_ip>:<subnet_mask>::<network_device>:none \
    --dest-karg-append nameserver=<nameserver_ip> \
    --dest-karg-append rd.neednet=1 -o \
    /root/rhcos-bootfiles/<node_name>-initramfs.s390x.img
```

注記

最初のブートの前に、クラスター内の各ノードの initramfs をカスタマイズし、PXE カーネルパラメーターを追加する必要があります。

3. **ignition.platform.id=metal** および **ignition.firstboot** を含むパラメーターファイルを作成します。

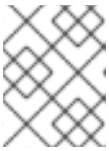
コントロールプレーンマシンのカーネルパラメーターファイルの例:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/dasda \(1)
ignition.firstboot ignition.platform.id=metal \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \(2)
coreos.inst.ignition_url=http://<http_server>/master.ign \(3)
ip=10.19.17.2::10.19.17.1:255.255.255.0::enbdd0:none nameserver=10.19.17.1 \
zfcp.allow_lun_scan=0 \(4)
rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
rd.zfcp=0.0.5677,0x600606680g7f0056,0x034F000000000000 \(5)
```

1. DASD タイプのディスクにインストールする場合は、**coreos.inst.install_dev=/dev/dasda** を追加します。FCP タイプのディスクの場合は、この値を省略します。
2. 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
3. Ignition 設定ファイルの場所を指定します。**master.ign** または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

4 FCP タイプのディスクにインストールする場合は、**zfcp.allow_lun_scan=0** を追加します。DASD タイプのディスクの場合は、この値を省略します。

5 DASD タイプのディスクにインストールする場合は、**rd.dasd=0.0.3490** に置き換えて DASD デバイスを指定します。



注記

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

関連情報

- Butane でのマシン設定の作成

18.7.12. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

独自にプロビジョニングする IBM Z® インフラストラクチャーに OpenShift Container Platform をインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を LPAR にインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、RHCOS ゲストマシンの再起動後に、OpenShift Container Platform ブートストラッププロセスが自動的に開始します。

マシンを作成するには、以下の手順を実行します。

前提条件

- 作成するマシンがアクセスできるプロビジョニングマシンで稼働している HTTP または HTTPS サーバー。
- セキュアブートを有効にする場合は、適切な Product Signing Key を取得し、IBM ドキュメントの Secure boot on IBM Z and IBM LinuxONE を確認した。

手順

1. プロビジョニングマシンで Linux にログインします。
2. RHCOS イメージミラー から Red Hat Enterprise Linux CoreOS (RHCOS) カーネル、 initramfs および rootfs ファイルを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な kernel、initramfs、および rootfs アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

注記

rootfs イメージは FCP および DASD の場合と同じです。

3. パラメーターファイルを作成します。以下のパラメーターは特定の仮想マシンに固有のものです。
   - **ip=** には、以下の 7 つのエントリーを指定します。
     i. マシンの IP アドレス。
     ii. 空の文字列。
     iii. ゲートウェイ。
     iv. ネットマスク。
     v. **hostname.domainname** 形式のマシンホストおよびドメイン名。この値を省略して、RHCOS に決定させるようにします。
     vi. ネットワークインターフェイス名。この値を省略して、RHCOS に決定させるようにします。
     vii. 静的 IP アドレスを使用する場合、 **none** を指定します。
   - **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
   - **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
   - オプション: セキュアブートを有効にするには、**coreos.inst.secure_ipl** を追加します。
   - DASD タイプのディスクへのインストールには、以下のタスクを実行します。
     i. **coreos.inst.install_dev=** には、**/dev/dasda** を指定します。
     ii. **rd.dasd=** を使用して、RHCOS がインストールされる DASD を指定します。
     iii. その他のパラメーターはすべて変更しません。
     ブートストラップマシンのパラメーターファイルのサンプル **bootstrap-0.parm**:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/dasda \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.
<architecture>.img \(1)
coreos.inst.ignition_url=http://<http_server>/bootstrap.ign \(2)
```

```
coreos.inst.secure_ipl \3
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.dasd=0.0.3490
```

1 起動する **kernel** と **initramfs** の **rootfs** アーティファクトの場所を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

2 Ignition 設定ファイルの場所を指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

3 オプション: セキュアブートを有効にするには、**coreos.inst.secure_ipl** を追加します。

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

- FCP タイプのディスクへのインストールには、以下のタスクを実行します。

  i. **rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して RHCOS がインストールされる FCP ディスクを指定します。マルチパスの場合、それぞれの追加のステップについてこのステップを繰り返します。

  注記

  複数のパスを使用してインストールする場合は、問題が発生する可能性があるため、後でではなくインストールの直後にマルチパスを有効にする必要があります。

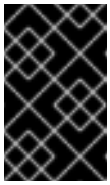
  ii. インストールデバイスを **coreos.inst.install_dev=/dev/disk/by-id/scsi-<serial_number>** として設定します。

  注記

  追加の LUN が NPIV で設定される場合は、FCP に **zfcp.allow_lun_scan=0** が必要です。CSI ドライバーを使用するために **zfcp.allow_lun_scan=1** を有効にする必要がある場合などには、各ノードが別のノードのブートパーティションにアクセスできないように NPIV を設定する必要があります。

  iii. その他のパラメーターはすべて変更しません。

  重要

  マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

  以下は、マルチパスが設定されたコンピュートノードのパラメーターファイルのサンプル **worker-1.parm** です。

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=/dev/disk/by-id/scsi-<serial_number> \
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.
<architecture>.img \
coreos.inst.ignition_url=http://<http_server>/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

4. FTP などを使用し、initramfs、kernel、パラメーターファイル、および RHCOS イメージを LPAR に転送します。FTP でファイルを転送して起動する方法については、LPAR へのインストール を参照してください。

5. マシンを起動します。

6. クラスター内の他のマシンについてこの手順を繰り返します。

18.7.12.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

18.7.12.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

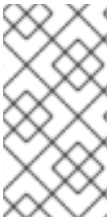
ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

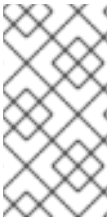
- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定

以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、options はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:em1,em2:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードを使用する場合の問題を回避するために、常にアクティブバックアップモードで **fail_over_mac=1** オプションを設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

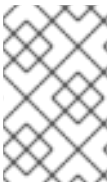
```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1**、**em2**) のコンマ区切りリストを表します。



注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

18.7.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

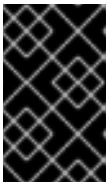
出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
```

```
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

18.7.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

18.7.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

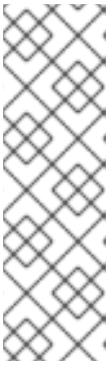
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

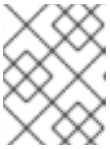
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

18.7.16. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                            4.16.0    True        False         False      40m
cluster-autoscaler                          4.16.0    True        False         False      37m
config-operator                             4.16.0    True        False         False      38m
```

```
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

18.7.16.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

18.7.16.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

18.7.16.2.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z® にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME           VERSION      AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16         True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

18.7.16.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

18.7.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
```

```
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

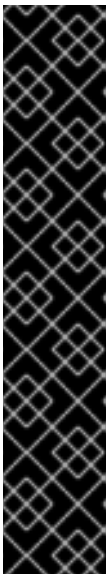
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running    0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running    0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace> 1
   ```

   1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

4. Cluster registration ページでクラスターを登録します。

検証

OpenShift Container Platform のブートストラッププロセス中にセキュアブートを有効にした場合は、次の検証手順が必要です。

1. 次のコマンドを実行してノードをデバッグします。

   ```
   $ oc debug node/<node_name>
   chroot /host
   ```

2. 次のコマンドを実行して、セキュアブートが有効になっていることを確認します。

   ```
   $ cat /sys/firmware/ipl/secure
   ```

   出力例

```
1 (1)
```

(1) セキュアブートが有効になっている場合、値は **1** です。有効になっていない場合は **0** です。

3. 次のコマンドを実行して、再 IPL 設定をリスト表示します。

```
# lsreipl
```

FCP ディスクの出力例

```
Re-IPL type: fcp
WWPN: 0x500507630400d1e3
LUN: 0x4001400e00000000
Device: 0.0.810e
bootprog: 0
br_lba: 0
Loadparm: ""
Bootparms: ""
clear: 0
```

DASD ディスクの出力例

```
for DASD output:
Re-IPL type: ccw
Device: 0.0.525d
Loadparm: ""
clear: 0
```

4. 次のコマンドを実行してノードをシャットダウンします。

```
sudo shutdown -h
```

5. Hardware Management Console (HMC) から LPAR からのブートを開始します。IBM ドキュメントの Initiating a secure boot from an LPAR を参照してください。

6. ノードが復帰したら、セキュアブートのステータスを再度確認します。

関連情報

- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH

18.7.18. 次のステップ

- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

- 必要に応じて、非接続クラスターの登録 を参照してください。

18.8. IBM Z および IBM LINUXONE のインストール設定パラメーター

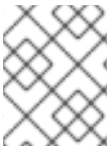
OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

このドキュメントは IBM Z® のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。

18.8.1. IBM Z で使用可能なインストール設定パラメーター

以下の表に、インストールプロセスの一部として設定できる必須、オプション、および IBM Z 固有のインストール設定パラメーターを示します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

18.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表18.114 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

18.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

- Red Hat OpenShift Networking OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 と IPv6 の両方のアドレスファミリーがサポートされます。

両方の IP アドレスファミリーを使用するようにクラスターを設定する場合は、次の要件を確認してください。

- どちらの IP ファミリーも、デフォルトゲートウェイに同じネットワークインターフェイスを使用する必要があります。
- 両方の IP ファミリーにデフォルトゲートウェイが必要です。
- すべてのネットワーク設定パラメーターに対して、IPv4 アドレスと IPv6 アドレスを同じ順序で指定する必要があります。たとえば、以下の設定では、IPv4 アドレスは IPv6 アドレスの前に記載されます。

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd00:10:128::/56
    hostPrefix: 64
  serviceNetwork:
  - 172.30.0.0/16
  - fd00:172:16::/112
```

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表18.115 ネットワークパラメーター

パラメーター	説明	値
`networking:`	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
`networking:` `  networkType:`	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。

パラメーター	説明	値
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

18.8.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表18.116 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列

パラメーター	説明	値
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**

パラメーター	説明	値
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**

パラメーター	説明	値
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。	**false** または **true**

パラメーター	説明	値
	重要  クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	

パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。  **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。

パラメーター	説明	値
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

第19章 IBM POWER SYSTEMS へのインストール

19.1. IBM POWER へのインストールの準備

19.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。

19.1.2. IBM Power に OpenShift Container Platform をインストールする方法の選択

以下の方法のいずれかを使用して、独自にプロビジョニングする IBM Power® インフラストラクチャーにクラスターをインストールできます。

- **クラスターの IBM Power® へのインストール**: OpenShift Container Platform を独自にプロビジョニングする IBM Power® インフラストラクチャーにインストールできます。
- **ネットワークが制限された環境での IBM Power® へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、独自にプロビジョニングする IBM Power® インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

19.2. クラスターの IBM POWER へのインストール

OpenShift Container Platform バージョン 4.16 では、独自にプロビジョニングする IBM Power® インフラストラクチャーにクラスターをインストールできます。

重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

19.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

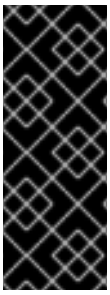
プロキシーを設定する場合は、このサイトリストも確認してください。

19.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

19.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

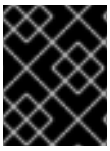
19.2.3.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表19.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。

ホスト	説明
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

19.2.3.2. クラスターインストールの最小リソース要件

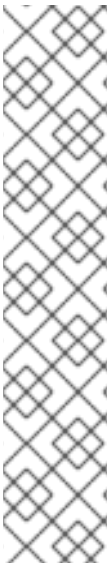
それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表19.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1 秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	2	16 GB	100 GB	300
コントロールプレーン	RHCOS	2	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform と Kubernetes は、ディスクのパフォーマンスの影響を受けるため、特にコントロールプレーンノードの etcd には、より高速なストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

19.2.3.3. IBM Power の最小要件

OpenShift Container Platform バージョン 4.16 は、以下の IBM® ハードウェアにインストールできます。

- IBM Power®9 または IBM Power®10 プロセッサーベースのシステム

注記

すべての IBM Power®8 モデル、IBM Power® AC922、IBM Power® IC922、および IBM Power® LC922 の RHCOS 機能のサポートは、OpenShift Container Platform 4.16 では非推奨です。Red Hat は、新しいハードウェアモデルを使用することを推奨します。

ハードウェア要件

- 複数の PowerVM サーバーにわたる 6 つの論理区画 (LPAR)

オペレーティングシステム要件

- IBM Power®9、または Power10 プロセッサーベースのシステムの 1 つのインスタンス

IBM Power® インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用の 3 つの LPAR
- OpenShift Container Platform コンピュートマシン用の 2 つの LPAR
- 一時 OpenShift Container Platform ブートストラップマシン用に 1 つの LPAR

IBM Power ゲスト仮想マシンのディスクストレージ

- ローカルストレージ、または vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 専用の物理アダプター、または SR-IOV 仮想機能
- 共有イーサネットアダプターを使用して仮想 I/O サーバーで利用可能
- IBM® vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 100GB / 16GB
- OpenShift Container Platform コンピュートマシン用に 100 GB / 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 100 GB / 16 GB

19.2.3.4. 推奨される IBM Power システム要件

ハードウェア要件

- 複数の PowerVM サーバーにわたる 6 つの LPAR

オペレーティングシステム要件

- IBM Power®9 または IBM Power®10 プロセッサーベースのシステムの 1 つのインスタンス

IBM Power® インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用の 3 つの LPAR
- OpenShift Container Platform コンピュートマシン用の 2 つの LPAR
- 一時 OpenShift Container Platform ブートストラップマシン用に 1 つの LPAR

IBM Power ゲスト仮想マシンのディスクストレージ

- ローカルストレージ、または vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 専用の物理アダプター、または SR-IOV 仮想機能
- 共有イーサネットアダプターを使用して仮想 I/O サーバーによって仮想化される
- IBM® vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 120 GB / 32 GB
- OpenShift Container Platform コンピュートマシン用に 120 GB / 32 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 120 GB / 16 GB

19.2.3.5. 証明書署名要求の管理

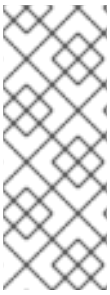
ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

19.2.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。



注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

19.2.3.6.1. DHCP を使用したクラスターノードのホスト名の設定

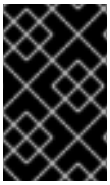
Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

19.2.3.6.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表19.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表19.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表19.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

19.2.3.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコード

で、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表19.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

19.2.3.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例19.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
```

```
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

```
-
```

例19.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1. Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。
2. Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。
3. ブートストラップマシンの逆引き DNS 解決を提供します。
4. 5. 6. コントロールプレーンマシンの逆引き DNS 解決を提供します。
7. 8. コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

19.2.3.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表19.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

    - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
    - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

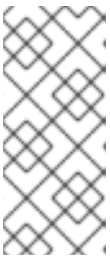
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表19.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



注記

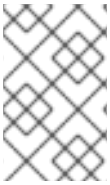
ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

19.2.3.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューショ

ンを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例19.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
```

```
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 5
    bind *:443
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:443 check inter 1s
    server compute1 compute1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 6
    bind *:80
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:80 check inter 1s
    server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

19.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用

の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

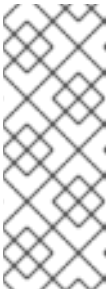
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

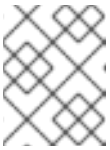
1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。
   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。



注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

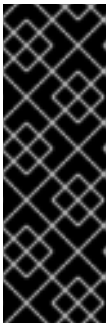
   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。



注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。



重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

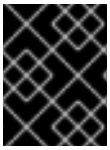
6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

19.2.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   注記

   出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

   random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1. Kubernetes 内部 API のレコード名を指定します。
2. Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

19.2.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

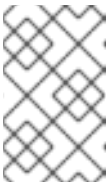
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

19.2.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

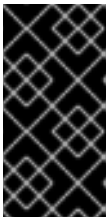
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

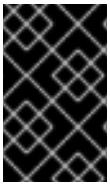
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

19.2.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

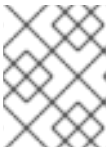
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

19.2.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

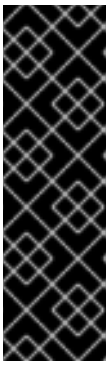
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

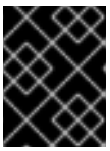
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Power® のインストール設定パラメーター

19.2.9.1. IBM Power のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture: ppc64le
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture: ppc64le
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
```

```
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths": ...}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッド (SMT) はサポートされていません。

(4) OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。

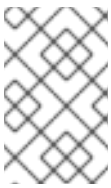
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

(7) クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

(8) DNS レコードに指定したクラスター名。

(9) Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

(10) それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Power® インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

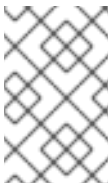
重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

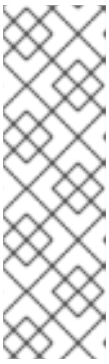
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

19.2.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

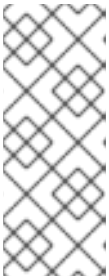
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする

trusted-ca-bundle 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

19.2.9.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定されるベアメタルクラスターに、ゼロコンピュートマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

19.2.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

19.2.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表19.9 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表19.10 defaultNetwork オブジェクト

フィールド	型	説明

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表19.11 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。

フィールド	型	説明
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表19.12 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表19.13 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表19.14 **policyAuditConfig** オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表19.15 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表19.16 gatewayConfig.ipv4 object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表19.17 gatewayConfig.ipv6 object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表19.18 ipsecConfig オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表19.19 kubeProxyConfig オブジェクト

フィールド	型	説明

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。 **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

19.2.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。Linux バージョンのインストールプログラム (アーキテクチャーポストフィックスなし) は、ppc64le でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

19.2.12. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Power® インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

ISO イメージまたはネットワーク PXE ブートを使用する手順を実行して RHCOS をマシンにインストールできます。

19.2.12.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

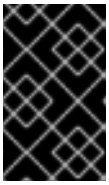
手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

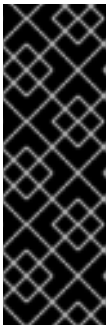
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

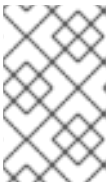
RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、**sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。



注記

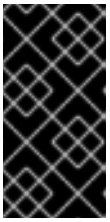
TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

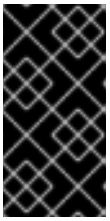
OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後、システムを再起動する必要があります。システムの再起動後、指定した Ignition 設定ファイルを適用します。
10. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

11. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

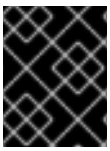
RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

19.2.12.1.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

19.2.12.1.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。



重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

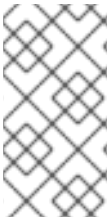
ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

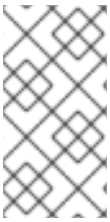
- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

  ```
  ip=::10.10.10.254::::
  ```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

  ```
  rd.route=20.20.20.0/24:20.20.20.254:enp2s0
  ```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

  ```
  ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
  vlan=enp2s0.100:enp2s0
  ```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

  ```
  ip=enp2s0.100:dhcp
  vlan=enp2s0.100:enp2s0
  ```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、options はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

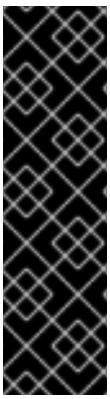
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:em1,em2:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合する



重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

オプション: **bond=** オプションを使用して、複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合できます。

各ノードで、次のタスクを実行する必要があります。

1. SR-IOV デバイスの管理 のガイダンスに従って、SR-IOV 仮想機能 (VF) を作成します。「仮想マシンへの SR-IOV ネットワークデバイスの接続」セクションの手順に従います。
2. ボンドを作成し、目的の VF をボンドに接続し、ネットワークボンディングの設定 のガイダンスに従って、ボンドリンクの状態を設定します。説明されている手順のいずれかに従って、結合を作成します。

次の例は、使用する必要がある構文を示しています。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は仮想機能 (VF) をカーネル内の既知の名前で表し、**ip link** コマンド (**eno1f0**、**eno2f0**) の出力に表示されます。**options** は結合オプションのコンマ区切りリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。
- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

19.2.12.2. PXE ブートを使用した RHCOS のインストール

PXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

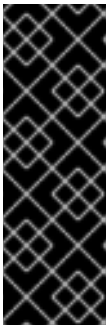
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+
(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



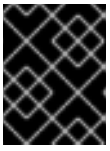
重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs**、**kernel**、および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE インストールを設定し、インストールを開始します。
以下の例で示されるご使用の環境のメニューエントリーを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
(2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

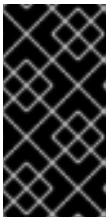
(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

7. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

9. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

10. クラスターのマシンの作成を続行します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

19.2.12.3. RHCOS のカーネル引数でのマルチパスの有効化

OpenShift Container Platform バージョン 4.16 では、インストール時に、プロビジョニングしたノードのマルチパスを有効にできます。RHCOS は、プライマリーディスクでのマルチパスをサポートします。マルチパス化により、ハードウェア障害に強力な耐障害性に利点が追加され、ホストの可用性が向上されます。

初回のクラスターの作成時に、カーネル引数をすべてのマスターまたはワーカーノードに追加しないといけない場合があります。カーネル引数をマスターまたはワーカーノードに追加するには、**MachineConfig** オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入することができます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. カーネル引数をワーカーまたコントロールプレーンノードに追加するかどうかを決定します。

- マシン設定ファイルを作成します。たとえば、**master** ラベルを追加し、マルチパスカーネル引数を指定するようクラスターに指示する **99-master-kargs-mpath.yaml** を作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "master"
  name: 99-master-kargs-mpath
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

3. ワーカーノードでマルチパスを有効にするには、以下を実行します。

   - マシン設定ファイルを作成します。たとえば、**worker** ラベルを追加し、マルチパスカーネル引数を指定するようクラスターに指示する **99-worker-kargs-mpath.yaml** を作成します。

     ```
     apiVersion: machineconfiguration.openshift.io/v1
     kind: MachineConfig
     metadata:
       labels:
         machineconfiguration.openshift.io/role: "worker"
       name: 99-worker-kargs-mpath
     spec:
       kernelArguments:
         - 'rd.multipath=default'
         - 'root=/dev/disk/by-label/dm-mpath-root'
     ```

     クラスターの作成を継続できます。

重要

マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

MPIO が失敗する場合は、bootlist コマンドを使用して、別の論理デバイス名でブートデバイスリストを更新します。このコマンドは、ブートリストを表示し、システムが通常モードで起動したときのブートデバイスを指定します。

a. ブートリストを表示し、システムが通常モードで起動した場合に使用可能なブートデバイスを指定するには、以下のコマンドを実行します。

   ```
   $ bootlist -m normal -o
   sda
   ```

b. 通常モードのブートリストを更新し、別のデバイス名を追加するには、以下のコマンドを実行します。

   ```
   $ bootlist -m normal -o /dev/sdc /dev/sdd /dev/sde
   sdc
   sdd
   sde
   ```

元のブートディスクパスがダウンすると、ノードは通常のブートデバイスリストに登録された別のデバイスから再起動します。

19.2.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

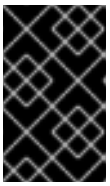
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

19.2.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

19.2.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

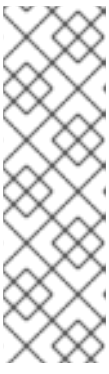
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

19.2.16. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                            4.16.0    True        False         False      40m
cluster-autoscaler                          4.16.0    True        False         False      37m
config-operator                             4.16.0    True        False         False      38m
```

```
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

19.2.16.1. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

19.2.16.1.1. IBM Power の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Power® にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION        AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry    4.16           True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

19.2.16.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

19.2.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

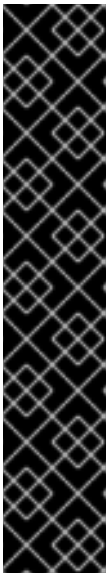
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver             apiserver-z25h4                             1/1     Running    0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
Running    0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. マルチパスを有効にするための追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

19.2.18. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

19.2.19. 次のステップ

- RHCOS でカーネル引数を使用してマルチパスを有効化 します。
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

19.3. ネットワークが制限された環境での IBM POWER へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、制限されたネットワーク内で、独自にプロビジョニングする IBM Power® インフラストラクチャーにクラスターをインストールできます。

重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

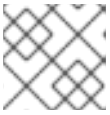
19.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ネットワークが制限された環境でインストールのミラーレジストリーを作成 し、お使いの OpenShift Container Platform のバージョンの **imageContentSources** データを取得している。
- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

重要

インストールメディアにアクセスできるマシンでインストール手順が実行されるようにします。

- 永続ストレージを OpenShift Data Foundation またはその他のサポートされているクラスター用ストレージプロトコルを使用してプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。



注記

プロキシーを設定する場合は、このサイトリストも確認してください。

19.3.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

19.3.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

19.3.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

19.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

19.3.4.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表19.20 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

19.3.4.2. クラスターインストールの最小リソース要件

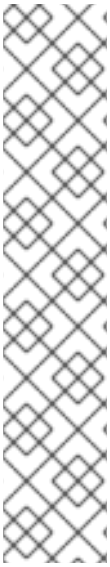
それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表19.21 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	2	16 GB	100 GB	300
コントロールプレーン	RHCOS	2	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform と Kubernetes は、ディスクのパフォーマンスの影響を受けるため、特にコントロールプレーンノードの etcd には、より高速なストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

19.3.4.3. IBM Power の最小要件

OpenShift Container Platform バージョン 4.16 は、以下の IBM® ハードウェアにインストールできます。

- IBM Power®9 または IBM Power®10 プロセッサーベースのシステム

注記

すべての IBM Power®8 モデル、IBM Power® AC922、IBM Power® IC922、および IBM Power® LC922 の RHCOS 機能のサポートは、OpenShift Container Platform 4.16 では非推奨です。Red Hat は、新しいハードウェアモデルを使用することを推奨します。

ハードウェア要件

- 複数の PowerVM サーバーにわたる 6 つの論理区画 (LPAR)

オペレーティングシステム要件

- IBM Power®9、または Power10 プロセッサーベースのシステムの 1 つのインスタンス

IBM Power® インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用の 3 つの LPAR
- OpenShift Container Platform コンピュートマシン用の 2 つの LPAR
- 一時 OpenShift Container Platform ブートストラップマシン用に 1 つの LPAR

IBM Power ゲスト仮想マシンのディスクストレージ

- ローカルストレージ、または vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 専用の物理アダプター、または SR-IOV 仮想機能
- 共有イーサネットアダプターを使用して仮想 I/O サーバーで利用可能
- IBM® vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 100GB / 16GB
- OpenShift Container Platform コンピュートマシン用に 100 GB / 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 100 GB / 16 GB

19.3.4.4. 推奨される IBM Power システム要件

ハードウェア要件

- 複数の PowerVM サーバーにわたる 6 つの LPAR

オペレーティングシステム要件

- IBM Power®9 または IBM Power®10 プロセッサーベースのシステムの 1 つのインスタンス

IBM Power® インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用の 3 つの LPAR
- OpenShift Container Platform コンピュートマシン用の 2 つの LPAR
- 一時 OpenShift Container Platform ブートストラップマシン用に 1 つの LPAR

IBM Power ゲスト仮想マシンのディスクストレージ

- ローカルストレージ、または vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 専用の物理アダプター、または SR-IOV 仮想機能
- 共有イーサネットアダプターを使用して仮想 I/O サーバーによって仮想化される
- IBM® vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 120 GB / 32 GB
- OpenShift Container Platform コンピュートマシン用に 120 GB / 32 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 120 GB / 16 GB

19.3.4.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

19.3.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

19.3.4.6.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

19.3.4.6.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表19.22 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表19.23 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表19.24 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

19.3.4.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表19.25 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

19.3.4.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例19.4 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
```

```
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例19.5 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
```

```
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

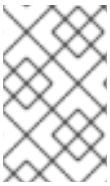
7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

19.3.4.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

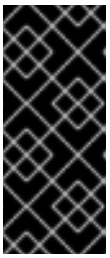


注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表19.26 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表19.27 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

19.3.4.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例19.6 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
```

```
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (5)
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (6)
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

(1) ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

(2) (4) ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

19.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。

a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

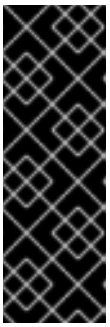
c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。



重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

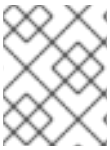
b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

19.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

```
api.ocp4.example.com.  604800 IN A 192.168.1.5
```

b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

出力例

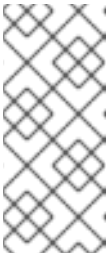
```
api-int.ocp4.example.com.  604800 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
```



注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

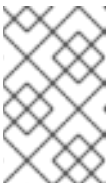
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

19.3.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

19.3.8. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Power® のインストール設定パラメーター

19.3.8.1. IBM Power のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
  architecture: ppc64le
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture: ppc64le
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
```

```
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
additionalTrustBundle: | (17)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (18)
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッド (SMT) はサポートされていません。

(4) OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。

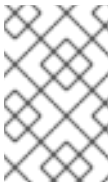
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

(7) クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

(8) DNS レコードに指定したクラスター名。

(9) Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Power® インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

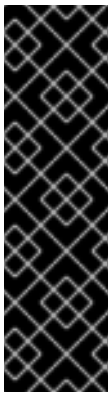
16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

18 リポジトリーのミラーリングに使用したコマンドの出力に従って、**imageContentSources** セクションを指定します。



重要

- **oc adm release mirror** コマンドを使用する場合は、**imageContentSources** セクションの出力を使用します。
- **oc mirror** コマンドを使用する場合は、コマンドの実行によって生成される **ImageContentSourcePolicy** ファイルの **repositoryDigestMirrors** セクションを使用します。
- **ImageContentSourcePolicy** は非推奨になりました。詳細は、**イメージレジストリーリポジトリーミラーリングの設定** を参照してください。

19.3.8.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

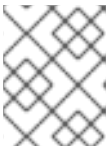
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

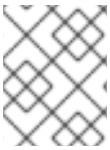
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

19.3.8.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定されるベアメタルクラスターに、ゼロコンピュートマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの

mastersSchedulable パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

19.3.9. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

19.3.9.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表19.28 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23

フィールド	型	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表19.29 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表19.30 **ovnKubernetesConfig** オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。  **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表19.31 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表19.32 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表19.33 **policyAuditConfig** オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表19.34 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

フィールド	型	説明
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表19.35 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表19.36 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表19.37 **ipsecConfig** オブジェクト

フィールド	型	説明

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表19.38 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	型	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

19.3.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。Linux バージョンのインストールプログラム (アーキテクチャーポストフィックスなし) は、ppc64le でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

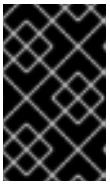
1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

19.3.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Power® インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

ISO イメージまたはネットワーク PXE ブートを使用する手順を実行して RHCOS をマシンにインストールできます。

19.3.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

   ```
   $ sha512sum <installation_directory>/bootstrap.ign
   ```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

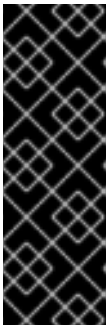
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



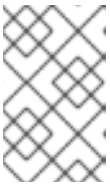
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。



注記

RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。

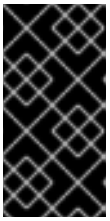
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後、システムを再起動する必要があります。システムの再起動後、指定した Ignition 設定ファイルを適用します。
10. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

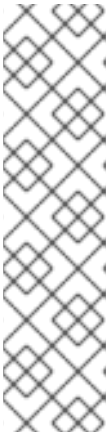
```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

11. 継続してクラスターの他のマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

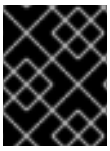
19.3.11.1.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

19.3.11.1.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、

RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。



重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

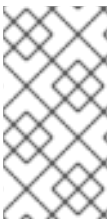
ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

  ```
  ip=::10.10.10.254::::
  ```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

  ```
  rd.route=20.20.20.0/24:20.20.20.254:enp2s0
  ```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、**options** はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合する

重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

オプション: **bond=** オプションを使用して、複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合できます。

各ノードで、次のタスクを実行する必要があります。

1. SR-IOV デバイスの管理 のガイダンスに従って、SR-IOV 仮想機能 (VF) を作成します。「仮想マシンへの SR-IOV ネットワークデバイスの接続」セクションの手順に従います。
2. ボンドを作成し、目的の VF をボンドに接続し、ネットワークボンディングの設定 のガイダンスに従って、ボンドリンクの状態を設定します。説明されている手順のいずれかに従って、結合を作成します。

次の例は、使用する必要がある構文を示しています。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は仮想機能 (VF) をカーネル内の既知の名前で表し、**ip link** コマンド (**eno1f0**、**eno2f0**) の出力に表示されます。**options** は結合オプションのコンマ区切りリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。
- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

19.3.11.2. PXE ブートを使用した RHCOS のインストール

PXE ブートを使用してマシンに RHCOS をインストールできます。

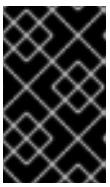
前提条件

- クラスターの Ignition 設定ファイルを作成している。

- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

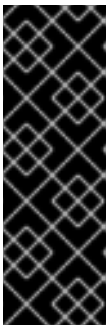
3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
```

```
"<url>/art/storage/releases/rhcos-4.16-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```



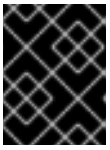
重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs**、**kernel**、および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE インストールを設定し、インストールを開始します。
以下の例で示されるご使用の環境のメニューエントリーを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

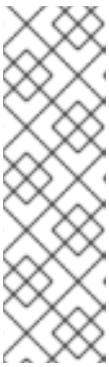
```
DEFAULT pxeboot
TIMEOUT 20
```

```
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
(2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

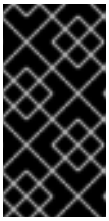
(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

7. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

9. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

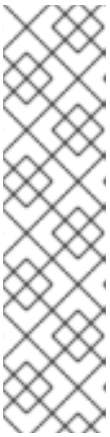
```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

10. クラスターのマシンの作成を続行します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

19.3.11.3. RHCOS のカーネル引数でのマルチパスの有効化

OpenShift Container Platform バージョン 4.16 では、インストール時に、プロビジョニングしたノードのマルチパスを有効にできます。RHCOS は、プライマリーディスクでのマルチパスをサポートします。マルチパス化により、ハードウェア障害に強力な耐障害性に利点が追加され、ホストの可用性が向上されます。

初回のクラスターの作成時に、カーネル引数をすべてのマスターまたはワーカーノードに追加しないといけない場合があります。カーネル引数をマスターまたはワーカーノードに追加するには、**MachineConfig** オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入することができます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. カーネル引数をワーカーまたはコントロールプレーンノードに追加するかどうかを決定します。

   - マシン設定ファイルを作成します。たとえば、**master** ラベルを追加し、マルチパスカーネル引数を指定するようクラスターに指示する **99-master-kargs-mpath.yaml** を作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "master"
  name: 99-master-kargs-mpath
```

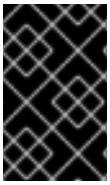
```
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

3. ワーカーノードでマルチパスを有効にするには、以下を実行します。

   - マシン設定ファイルを作成します。たとえば、**worker** ラベルを追加し、マルチパスカーネル引数を指定するようクラスターに指示する **99-worker-kargs-mpath.yaml** を作成します。

     ```
     apiVersion: machineconfiguration.openshift.io/v1
     kind: MachineConfig
     metadata:
       labels:
         machineconfiguration.openshift.io/role: "worker"
       name: 99-worker-kargs-mpath
     spec:
       kernelArguments:
         - 'rd.multipath=default'
         - 'root=/dev/disk/by-label/dm-mpath-root'
     ```

     クラスターの作成を継続できます。



重要

マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

MPIO が失敗する場合は、bootlist コマンドを使用して、別の論理デバイス名でブートデバイスリストを更新します。このコマンドは、ブートリストを表示し、システムが通常モードで起動したときのブートデバイスを指定します。

a. ブートリストを表示し、システムが通常モードで起動した場合に使用可能なブートデバイスを指定するには、以下のコマンドを実行します。

   ```
   $ bootlist -m normal -o
   sda
   ```

b. 通常モードのブートリストを更新し、別のデバイス名を追加するには、以下のコマンドを実行します。

   ```
   $ bootlist -m normal -o /dev/sdc /dev/sdd /dev/sde
   sdc
   sdd
   sde
   ```

   元のブートディスクパスがダウンすると、ノードは通常のブートデバイスリストに登録された別のデバイスから再起動します。

19.3.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにイ

ンストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

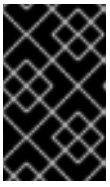
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

19.3.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラス

ターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

19.3.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

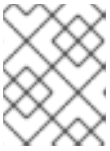
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

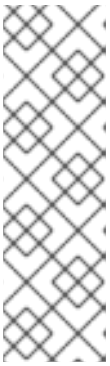
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

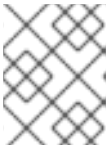
6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
```

```
master-2  Ready    master  74m  v1.29.4
worker-0  Ready    worker  11m  v1.29.4
worker-1  Ready    worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

19.3.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
```

```
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

19.3.15.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

19.3.15.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

19.3.15.2.1. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、Image Registry Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **managementState** Image Registry Operator 設定を **Removed** から **Managed** に変更します。以下に例を示します。

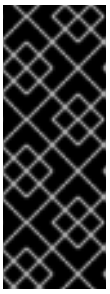
```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"managementState":"Managed"}}'
```

19.3.15.2.2. IBM Power の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Power® にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。



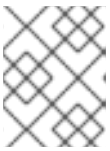
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

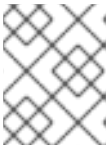
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME           VERSION      AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16         True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

19.3.15.2.3. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

19.3.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
```

```
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

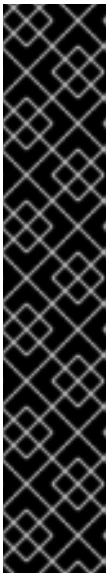
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running    0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running    0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace> 1
   ```

   1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. マルチパスを有効にするための追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

4. Cluster registration ページでクラスターを登録します。

19.3.17. 次のステップ

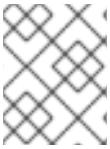
- RHCOS でカーネル引数を使用してマルチパスを有効化 します。
- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。

19.4. IBM POWER のインストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

19.4.1. IBM Power で使用可能なインストール設定パラメーター

以下の表に、インストールプロセスの一部として設定できる必須、オプション、および IBM Power 固有のインストール設定パラメーターを示します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

19.4.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表19.39 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**bare metal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

19.4.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

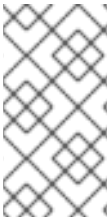
- Red Hat OpenShift Networking OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 と IPv6 の両方のアドレスファミリーがサポートされます。

両方の IP アドレスファミリーを使用するようにクラスターを設定する場合は、次の要件を確認してください。

- どちらの IP ファミリーも、デフォルトゲートウェイに同じネットワークインターフェイスを使用する必要があります。
- 両方の IP ファミリーにデフォルトゲートウェイが必要です。
- すべてのネットワーク設定パラメーターに対して、IPv4 アドレスと IPv6 アドレスを同じ順序で指定する必要があります。たとえば、以下の設定では、IPv4 アドレスは IPv6 アドレスの前に記載されます。

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
```

```
  hostPrefix: 23
- cidr: fd00:10:128::/56
  hostPrefix: 64
serviceNetwork:
- 172.30.0.0/16
- fd00:172:16::/112
```



注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表19.40 ネットワークパラメーター

パラメーター	説明	値
`networking:`	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
`networking:` `  networkType:`	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
`networking:` `  clusterNetwork:`	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 `networking:` `  clusterNetwork:` `  - cidr: 10.128.0.0/14` `  hostPrefix: 23`
`networking:` `  clusterNetwork:` `  cidr:`	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。

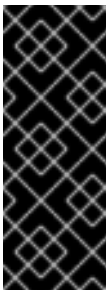
パラメーター	説明	値
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

19.4.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表19.41 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。

パラメーター	説明	値
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。

パラメーター	説明	値
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]

パラメーター	説明	値
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。 **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。 **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。 **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

第20章 IBM POWER VIRTUAL SERVER へのインストール

20.1. IBM POWER VIRTUAL SERVER へのインストールの準備

このセクションに記載されているインストールワークフローは、IBM Power® Virtual Server インフラストラクチャー環境を対象としています。

20.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。

20.1.2. IBM Power Virtual Server に OpenShift Container Platform をインストールするための要件

IBM Power® Virtual Server に OpenShift Container Platform をインストールする前に、サービスアカウントを作成し、IBM Cloud® アカウントを設定する必要があります。アカウントの作成、DNS およびサポートされる IBM Power® Virtual Server 領域の設定の詳細は、IBM Cloud® アカウントの設定 を参照してください。

クラスターを IBM Power® Virtual Server にインストールする場合は、クラウド認証情報を手動で管理する必要があります。これは、クラスターをインストールする前に、手動モードの Cloud Credential Operator (CCO) を設定して実行します。

20.1.3. IBM Power Virtual Server に OpenShift Container Platform をインストールする方法の選択

インストーラーがプロビジョニングしたインフラストラクチャーを使用して、OpenShift Container Platform を IBM Power® Virtual Server にインストールできます。このプロセスでは、インストールプログラムを使用して、クラスターの基盤となるインフラストラクチャーをプロビジョニングします。現時点では、ユーザーがプロビジョニングしたインフラストラクチャーを使用した IBM Power® Virtual Server への OpenShift Container Platform のインストールはサポートされていません。

インストーラーがプロビジョニングしたインストールプロセスの詳細については、インストールプロセス を参照してください。

20.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下のいずれかの方法を使用して、OpenShift Container Platform インストールプログラムによってプロビジョニングされた IBM Power® Virtual Server インフラストラクチャーにクラスターをインストールできます。

- **IBM Power® Virtual Server へのカスタマイズされたクラスターのインストール**: インストールプログラムがプロビジョニングする IBM Power® Virtual Server インフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の多くのカスタマイズオプションは、インストール後 に利用できます。
- **IBM Power® Virtual Server 上のクラスターを既存の VPC にインストールする**: IBM Power® Virtual Server 上の OpenShift Container Platform を既存の Virtual Private Cloud (VPC) にインストールできます。このインストール方法は、新規アカウントまたはインフラストラクチャーを作成する際の制限など、会社のガイドラインによる制約がある場合に使用できます。

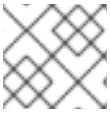
- プライベートクラスターの IBM Power® Virtual Server へのインストール：プライベートクラスターを IBM Power® Virtual Server にインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

- 制限されたネットワーク内の IBM Power® Virtual Server へのクラスターのインストール：インストールリリースコンテンツの内部ミラーを使用して、インストーラーがプロビジョニングしたインフラストラクチャー上の IBM Power® Virtual Server に OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。

20.1.4. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報を Kubernetes カスタムリソース定義 (CRD) として管理します。IBM Power® Virtual Server にクラスターをインストールするには、インストールプロセスの一部として CCO を **manual** モードに設定する必要があります。

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 & **lt;rhel_version&** gt; について、ホストが使用する Red Hat Enterprise Linux (RHEL)のバージョンに対応する値を指定します。値が指定されていない場合、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストにこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストにこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud    Manage credentials objects for IBM Cloud
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

関連情報

- API キーのローテーション

20.1.5. 次のステップ

- IBM Cloud® アカウントの設定

20.2. IBM CLOUD アカウントの設定

OpenShift Container Platform をインストールする前に、IBM Cloud® アカウントを設定する必要があります。

20.2.1. 前提条件

- サブスクリプションのある IBM Cloud® アカウントを持っている。OpenShift Container Platform を無料または試用版の IBM Cloud® アカウントにインストールすることはできません。

20.2.2. IBM Power Virtual Server のクォータと制限

OpenShift Container Platform クラスターは、いくつかの IBM Cloud® および IBM Power® Virtual Server コンポーネントを使用しており、デフォルトのクォータと制限は、OpenShift Container Platform クラスターをインストールする能力に影響します。特定のクラスター設定を使用する場合、特定のリージョンにクラスターをデプロイするか、アカウントから複数のクラスターを実行する場合は、IBM Cloud® アカウントに追加のリソースを要求する必要がある場合があります。

デフォルトの IBM Cloud® クォータとサービス制限の包括的なリストについては、IBM Cloud® のドキュメント Quotas and service limits を参照してください。

仮想プライベートクラウド
各 OpenShift Container Platform クラスターは、独自の Virtual Private Cloud (VPC) を作成します。リージョンあたりの VPC のデフォルトのクォータは 10 です。10 個の VPC を作成した場合は、インストールを試行する前にクォータを増やす必要があります。

アプリケーションロードバランサー
デフォルトでは、各クラスターは 2 つのアプリケーションロードバランサー (ALB) を作成します。

- コントロールプレーン API サーバーの内部ロードバランサー
- コントロールプレーン API サーバー用の外部ロードバランサー

追加の **LoadBalancer** サービスオブジェクトを作成して、追加の ALB を作成できます。VPC ALB のデフォルトのクォータは、リージョンごとに 50 です。50 を超える ALB を使用するには、このクオータを増やす必要があります。

VPC ALB がサポートされています。Classic ALB は、IBM Power® Virtual Server ではサポートされていません。

Transit Gateway
各 OpenShift Container Platform クラスターは、VPC との通信を可能にするために、独自の Transit Gateway を作成します。アカウントあたりの Transit Gateway のデフォルトのクォータは 10 です。10 個の Transit Gateway を作成済みの場合は、インストールを試行する前にクォータを増やす必要があります。

動的ホスト設定プロトコルサービス
IBM Power® Virtual Server インスタンスごとに Dynamic Host Configuration Protocol (DHCP) サービスは 1 つまでという制限があります。

ネットワーク
ネットワークの制限により、IPI を介してインストールされる OpenShift クラスターは、アカウントごとのゾーンごとに 1 つまでという制限があります。これは設定できません。

Virtual Server Instances
デフォルトでは、クラスターは次のリソースを使用してサーバーインスタンスを作成します。

- 0.5 CPU
- 32 GB RAM
- システムタイプ: **s922**
- プロセッサーのタイプ: **uncapped**、**shared**
- ストレージ階層: **Tier-3**

次のノードが作成されます。

- 1 台のブートストラップマシン。インストール完了後に削除されます。
- 3 つのコントロールプレーンノード。
- 3 つのコンピュートノード

詳細は、IBM Cloud® 資料の Power Systems 仮想サーバーの作成 を参照してください。

20.2.3. DNS 解決の設定

DNS 解決の設定方法は、インストールする OpenShift Container Platform クラスターのタイプによって異なります。

- パブリッククラスターをインストールする場合は、IBM Cloud® Internet Services (CIS) を使用します。
- プライベートクラスターをインストールする場合は、IBM Cloud® DNS サービス (DNS サービス) を使用します。

20.2.4. DNS 解決のための IBM Cloud Internet Services の使用

インストールプログラムは、IBM Cloud® Internet Services (CIS) を使用してクラスター DNS 解決を設定し、パブリッククラスターの名前検索を提供します。

注記

この製品は IPv6 をサポートしていないため、デュアルスタックまたは IPv6 環境は使用できません。

クラスターと同じアカウントの CIS にドメインゾーンを作成する必要があります。また、ゾーンがドメインに対して権限を持っていることを確認する必要があります。これは、root ドメインまたはサブドメインを使用して行うことができます。

前提条件

- IBM Cloud® CLI がインストールされている。
- 既存のドメインとレジストラがあります。詳細については、IBM® の ドキュメント を参照してください。

手順

1. クラスターで使用する CIS インスタンスを作成します。

a. CIS プラグインをインストールします。

```
$ ibmcloud plugin install cis
```

b. CLI を使用して IBM Cloud® にログインします。

```
$ ibmcloud login
```

c. CIS インスタンスを作成します。

```
$ ibmcloud cis instance-create <instance_name> standard 1
```

1 CIS がクラスターサブドメインとその DNS レコードを管理するには、少なくとも **Standard** プランが必要です。

2. 既存のドメインを CIS インスタンスに接続します。

a. CIS のコンテキストインスタンスを設定します。

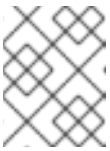
```
$ ibmcloud cis instance-set <instance_CRN> 1
```

1 インスタンス CRN (クラウドリソース名)。例: **ibmcloud cis instance-set crn:v1:bluemix:public:power-iaas:osa21:a/65b64c1f1c29460d8c2e4bbfbd893c2c:c09233ac-48a5-4ccb-a051-d1cfb3fc7eb5::**

b. CIS のドメインを追加します。

```
$ ibmcloud cis domain-add <domain_name> 1
```

1 完全修飾ドメイン名。設定する予定に応じて、ドメイン名として root ドメインまたはサブドメインのいずれかの値を使用できます。



注記

root ドメインは、**openshiftcorp.com** の形式を使用します。サブドメインは、**clusters.openshiftcorp.com** の形式を使用します。

3. CIS Web コンソール を開き、**Overview** ページに移動して、CIS ネームサーバーをメモします。これらのネームサーバーは、次のステップで使用されます。

4. ドメインのレジストラーまたは DNS プロバイダーでドメインまたはサブドメインのネームサーバーを設定します。詳細については、IBM Cloud® の ドキュメント を参照してください。

20.2.5. IBM Cloud IAM ポリシーと API キー

OpenShift Container Platform を IBMCloud® アカウントにインストールするには、インストールプログラムに IAM API キーが必要です。これにより、IBM Cloud® サービス API にアクセスするための認証と認証が提供されます。必要なポリシーを含む既存の IAM API キーを使用するか、新しいポリシーを作成できます。

IBM Cloud® IAM の概要は、IBM Cloud® の ドキュメント を参照してください。

20.2.5.1. 前提条件のパーミッション

表20.1 前提条件のパーミッション

ロール	アクセス
viewer、Operator、Editor、Administrator、Reader、Writer、Manager	<resource_group> リソースグループ内のインターネットサービス
Viewer、Operator、Editor、Administrator、User API key creator、Service ID creator	IAM Identity Service サービス
viewer、Operator、Administrator、Editor、Reader、Writer、Manager、Console Administrator	<resource_group> リソースグループの VPC Infrastructure Services サービス
Viewer	Resource Group: リソースグループ自体を表示するためのアクセス権。リソースタイプは **Resource group** と等しく、値は <your_resource_group_name> である必要があります。

20.2.5.2. Cluster-creation パーミッション

表20.2 Cluster-creation パーミッション

ロール	アクセス
Viewer	<resource_group> （チーム用に作成されたリソースグループ）
viewer、Operator、Editor、Reader、Writer、Manager	Default リソースグループ内のすべてのアイデンティティーおよび IAM 対応サービス
Viewer、Reader	Internet Services サービス
viewer、Operator、Reader、Writer、Manager、Content Reader、Object Reader、Object Writer、Editor	Cloud Object Storage サービス
Viewer	デフォルトのリソースグループ: リソースタイプは **Resource group** と等しく、値は **Default** である必要があります。アカウント管理者がアカウントのデフォルトのリソースグループをデフォルト以外に変更した場合は、代わりにその値を使用してください。
viewer、Operator、Editor、Reader、Manager	<resource_group> リソースグループの IBM Power® Virtual Server サービスのワークスペース

ロール	アクセス
viewer、Operator、Editor、Reader、Writer、Manager、Administrator	<resource_group> リソースグループのインターネットサービスサービス: CIS 機能スコープ文字列が信頼性に等しい
Viewer、Operator、Editor	Transit Gateway サービス
viewer、Operator、Editor、Administrator、Reader、Writer、Manager、Console Administrator	VPC Infrastructure Services サービス <resource_group> リソースグループ

20.2.5.3. アクセスポリシーの割り当て

IBM Cloud® IAM では、アクセスポリシーをさまざまなサブジェクトにアタッチできます。

- アクセスグループ (推奨)
- サービス ID
- User

推奨される方法は、アクセスグループ で IAM アクセスポリシーを定義することです。これにより、OpenShift Container Platform に必要なすべてのアクセスを整理し、ユーザーとサービス ID をこのグループにオンボードできます。必要に応じて、ユーザーとサービス ID に直接アクセスを割り当てることもできます。

20.2.5.4. API キーの作成

IBM Cloud® アカウントのユーザー API キーまたはサービス ID API キーを作成する必要があります。

前提条件

- 必要なアクセスポリシーを IBM Cloud® アカウントに割り当てている。
- IAM アクセスポリシーをアクセスグループまたはその他の適切なリソースにアタッチしている。

手順

- IAM アクセスポリシーの定義方法に応じて、API キーを作成します。
  たとえば、アクセスポリシーをユーザーに割り当てた場合は、ユーザー API キー を作成する必要があります。アクセスポリシーをサービス ID に割り当てた場合は、サービス ID API キー を作成する必要があります。アクセスポリシーがアクセスグループに割り当てられている場合は、どちらの API キータイプも使用できます。IBM Cloud® API キーの詳細は、Understanding API keys を参照してください。

20.2.6. サポートされている IBM Power Virtual Server のリージョンとゾーン

OpenShift Container Platform クラスターを以下のリージョンにデプロイできます。

- **dal** (アメリカ、ダラス)

    - **dal10**
    - **dal12**
- **eu-de** (Frankfurt, Germany)
    - **eu-de-1**
    - **eu-de-2**
- **lon** (ロンドン、英国)
    - **lon04**
- **mad** (Madrid, Spain)
    - **mad02**
    - **mad04**
- **osa** (大阪、日本)
    - **osa21**
- **sao** (サンパウロ、ブラジル)
    - **sao01**
    - **sao04**
- **syd** (オーストラリア、シドニー)
    - **syd04**
- **wdc** (Washington DC, USA)
    - **wdc06**
    - **wdc07**

オプションで、インストーラーが VPC コンポーネントを作成する IBM Cloud® リージョンを指定できます。IBM Cloud® でサポートされるリージョンは次のとおりです。

- **us-south**
- **eu-de**
- **eu-es**
- **eu-gb**
- **jp-osa**
- **au-syd**
- **br-sao**
- **ca-tor**
- **jp-tok**

20.2.7. 次のステップ

- IBM Power® Virtual Server ワークスペースの作成

20.3. IBM POWER VIRTUAL SERVER ワークスペースの作成

20.3.1. IBM Power Virtual Server ワークスペースの作成

IBM Power® Virtual Server ワークスペースを作成するには、次の手順を使用します。

手順

1. IBM Power® Virtual Server ワークスペースを作成するには、IBM Cloud® 資料の IBM Power® Virtual Server の作成 のステップ 1 からステップ 5 を実行します。
2. プロビジョニングが完了したら、次のコマンドを入力して、新しいワークスペースの 32 文字の英数字の Globally Unique Identifier (GUID) を取得します。

```
$ ibmcloud resource service-instance <workspace name>
```

20.3.2. 次のステップ

- カスタマイズを使用した IBM Power® Virtual Server へのクラスターのインストール

20.4. カスタマイズを使用した IBM POWER VIRTUAL SERVER へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、インストールプログラムが IBM Power Virtual Server 上にプロビジョニングするインフラストラクチャーに、カスタマイズしたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

20.4.1. 前提条件

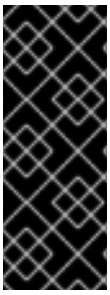
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように IBM Cloud® アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細は、Cloud Credential Operator ユーティリティーの設定 を参照してください。

20.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

20.4.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

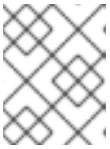
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

20.4.4. インストールプログラムの取得

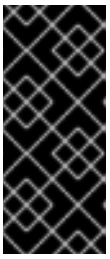
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

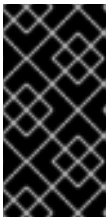
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

20.4.5. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

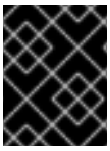
前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IBMCLOUD_API_KEY=<api_key>
```



重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

20.4.6. インストール設定ファイルの作成

インストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

注記

古い設定の再利用を回避するために、**~/.powervs** ディレクトリーは必ず削除してください。以下のコマンドを実行します。

```
$ rm -rf ~/.powervs
```

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットのプラットフォームとして **powervs** を選択します。

iii. クラスターをデプロイするリージョンを選択します。

iv. クラスターをデプロイするゾーンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

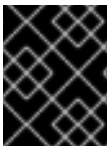
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- IBM Power® Virtual Server のインストール設定パラメーター

20.4.6.1. IBM Power Virtual Server 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com
compute: 1 2
- architecture: ppc64le
  hyperthreading: Enabled 3
  name: worker
  platform:
    powervs:
      smtLevel: 8 4
  replicas: 3
```

```
controlPlane: 5 6
  architecture: ppc64le
  hyperthreading: Enabled 7
  name: master
  platform:
    powervs:
      smtLevel: 8 8
  replicas: 3
metadata:
  creationTimestamp: null
  name: example-cluster-name
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.0.0/24
  networkType: OVNKubernetes 9
  serviceNetwork:
  - 172.30.0.0/16
platform:
  powervs:
    userID: ibm-user-id
    region: powervs-region
    zone: powervs-zone
    powervsResourceGroup: "ibmcloud-resource-group" 10
    serviceInstanceGUID: "powervs-region-service-instance-guid"
    vpcRegion : vpc-region
publish: External
pullSecret: '{"auths": ...}' 11
sshKey: ssh-ed25519 AAAA... 12
```

1 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

2 6 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。現在、どちらのセクションでも単一のマシンプールを定義していますが、OpenShift Container Platform がインストール中に複数のコンピューティングプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

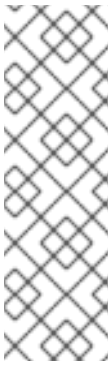
3 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 8 smtLevel は、コントロールプレーンとコンピュートマシンに設定する SMT のレベルを指定します。サポートされている値は、1、2、4、8、**'off'** および **'on'** です。デフォルト値は 8 です。smtLevel を **'off'** にすると、SMT がオフに設定されます。smtlevel を **'on'** にすると、クラスター

ノードで SMT がデフォルト値 8 に設定されます。



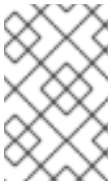
注記

同時マルチスレッド (SMT) またはハイパースレッディングが有効でない場合、1 vCPU が 1 つの物理コアに相当します。有効な場合、(コアあたりのスレッド数 x ソケットあたりのコア数) x ソケット数という式で合計 vCPU が計算されます。smtLevel はコアあたりのスレッド数を制御します。SMT レベルが低い場合、クラスターノードをデプロイするときに追加の割り当てコアが必要になる場合があります。これを行うには、**install-config.yaml** ファイルの **processors** パラメーターを、OpenShift Container Platform を正常にデプロイするための要件を満たす適切な値に設定します。

9 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

10 既存のリソースグループの名前。

11 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

20.4.6.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

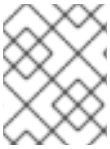
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

20.4.7. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

   ```
   apiVersion: v1
   baseDomain: cluster1.example.com
   credentialsMode: Manual (1)
   compute:
   - architecture: ppc64le
     hyperthreading: Enabled
   ```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
  --credentials-requests-dir=<path_to_credential_requests_directory> \ 1
  --name=<cluster_name> \ 2
  --output-dir=<installation_directory> \ 3
  --resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

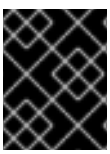
```
$ grep resourceGroup <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

20.4.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

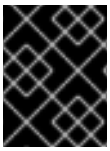
```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

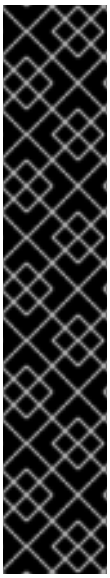


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

20.4.9. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

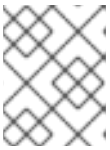
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

20.4.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

20.4.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 し

て、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

20.4.12. 次のステップ

- クラスターのカスタマイズ
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。

20.5. IBM POWER VIRTUAL SERVER 上のクラスターを既存の VPC にインストールする

OpenShift Container Platform バージョン 4.16 では、IBM Cloud® 上の既存の Virtual Private Cloud (VPC) にクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

20.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように IBM Cloud® アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細は、Cloud Credential Operator ユーティリティーの設定 を参照してください。

20.5.2. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、既存の IBM® Virtual Private Cloud (VPC) を使用してクラスターをデプロイできます。

インストールプログラムは既存のサブネットにある他のコンポーネントを認識できないため、サブネットの CIDR などを選択できません。クラスターをインストールするサブネットのネットワークを設定する必要があります。

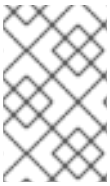
20.5.2.1. VPC を使用するための要件

クラスターをインストールする前に、既存の VPC およびそのサブネットを適切に設定する必要があります。このシナリオでは、インストールプログラムは VPC または VPC サブネットを作成しません。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化します。
- サブネットのルートテーブルを設定します。

- DHCP などの VPC オプションの設定



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

20.5.2.2. VPC 検証

VPC とすべてのサブネットは、既存のリソースグループ内にある必要があります。クラスターはこのリソースグループにデプロイされます。

インストールの一環として、**install-config.yaml** ファイルで以下を指定します。

- リソースグループの名前
- VPC の名前
- VPC サブネットの名前

指定したサブネットが適切であることを確認するために、インストールプログラムは、指定したすべてのサブネットが存在することを確認します。

注記

サブネット ID はサポートされていません。

20.5.2.3. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- ICMP Ingress はネットワーク全体に許可されます。
- TCP ポート 22 入力 (SSH) はネットワーク全体で許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

20.5.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

20.5.4. クラスターノードの SSH アクセス用のキーペアの生成

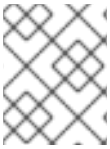
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

20.5.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

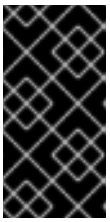
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

20.5.6. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IBMCLOUD_API_KEY=<api_key>
```

重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

20.5.7. インストール設定ファイルの作成

インストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- IBM Power® Virtual Server のインストール設定パラメーター

20.5.7.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表20.3 最小リソース要件

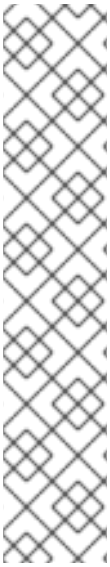
マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、

特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

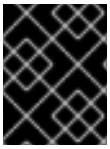
プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

20.5.7.2. IBM Power Virtual Server 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com
compute: 1 2
- architecture: ppc64le
  hyperthreading: Enabled 3
  name: worker
  platform:
    powervs:
```

```
      smtLevel: 8 4
  replicas: 3
controlPlane: 5 6
  architecture: ppc64le
  hyperthreading: Enabled 7
  name: master
  platform:
    powervs:
      smtLevel: 8 8
  replicas: 3
metadata:
  creationTimestamp: null
  name: example-cluster-existing-vpc
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.0.0/24
  networkType: OVNKubernetes 10
  serviceNetwork:
  - 172.30.0.0/16
platform:
  powervs:
    userID: ibm-user-id
    powervsResourceGroup: "ibmcloud-resource-group"
    region: powervs-region
    vpcRegion : vpc-region
    vpcName: name-of-existing-vpc 11
    vpcSubnets: 12
    - powervs-region-example-subnet-1
    zone: powervs-zone
    serviceInstanceGUID: "powervs-region-service-instance-guid"
credentialsMode: Manual
publish: External 13
pullSecret: '{"auths": ...}' 14
fips: false
sshKey: ssh-ed25519 AAAA... 15
```

1 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

2 6 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。現在、両方のセクションで単一のマシンプールが定義されています。1 つのコントロールプレーンプールのみが使用されます。

3 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

4 8 smtLevel は、コントロールプレーンとコンピュートマシンに設定する SMT のレベルを指定します。サポートされている値は、1、2、4、8、**'off'** および **'on'** です。デフォルト値は 8 です。

smtLevel を **'off'** にすると、SMT がオフに設定されます。smtlevel を **'on'** にすると、クラスターノードで SMT がデフォルト値 8 に設定されます。

注記

同時マルチスレッド (SMT) またはハイパースレッディングが有効でない場合、1 vCPU が 1 つの物理コアに相当します。有効な場合、(コアあたりのスレッド数 x ソケットあたりのコア数) x ソケット数という式で合計 vCPU が計算されます。smtLevel はコアあたりのスレッド数を制御します。SMT レベルが低い場合、クラスターノードをデプロイするときに追加の割り当てコアが必要になる場合があります。これを行うには、**install-config.yaml** ファイルの **processors** パラメーターを、OpenShift Container Platform を正常にデプロイするための要件を満たす適切な値に設定します。

9 マシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれている必要があります。

10 インストールするクラスターネットワークプラグイン。サポートされている値は **OVNKubernetes** です。

11 既存 VPC の名前を指定します。

12 既存の VPC サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。リージョン内の各アベイラビリティーゾーンのサブネットを指定します。

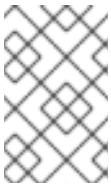
13 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法を指定します。

14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

15 クラスター内のマシンにアクセスするために使用する **sshKey** 値を指定します。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

20.5.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーを

バイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

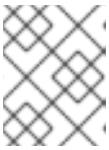
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップ

を参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

20.5.8. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual 1
```

```
compute:
- architecture: ppc64le
  hyperthreading: Enabled
```

1 この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ openshift-install create manifests --dir <installation_directory>
```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

2 **install-config.yaml** ファイルの場所を指定します。

3 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
```

```
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
  --credentials-requests-dir=<path_to_credential_requests_directory> \ 1
  --name=<cluster_name> \ 2
  --output-dir=<installation_directory> \ 3
  --resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

```
$ grep resourceGroup <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

20.5.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

20.5.10. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

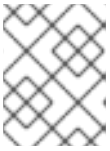
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

20.5.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

20.5.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

20.5.13. 次のステップ

- クラスターのカスタマイズ
- オプション: リモートヘルスレポートのオプトアウト

20.6. IBM POWER VIRTUAL SERVER へのプライベートクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、既存の VPC および IBM Power® Virtual Server Workspace にプライベートクラスターをインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

20.6.1. 前提条件

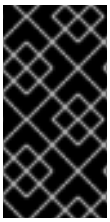
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように IBM Cloud® アカウントを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細は、Cloud Credential Operator ユーティリティーの設定 を参照してください。

20.6.2. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラス

ターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。



重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

プライベートクラスターをデプロイするには、以下を行う必要があります。

- 要件を満たす既存のネットワークを使用します。
- IBM Cloud® DNS Services を使用して DNS ゾーンを作成し、それをクラスターの基本ドメインとして指定します。詳しくは、IBM Cloud® DNS サービスを使用して DNS 解決を設定するを参照してください。
- 以下にアクセスできるマシンからデプロイ。
  - プロビジョニングするクラウドの API サービス。
  - プロビジョニングするネットワーク上のホスト。
  - インストールメディアを取得するインターネット。

これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

20.6.3. IBM Power Virtual Server のプライベートクラスター

IBM Power® Virtual Server 上にプライベートクラスターを作成するには、クラスターをホストするための既存のプライベート Virtual Private Cloud (VPC) とサブネットを提供する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

クラスターは、IBM Cloud® API にアクセスするために引き続きインターネットにアクセスする必要があります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックネットワークロードバランサー
- クラスターの **baseDomain** に一致するパブリック DNS ゾーン

また、**baseDomain** と一致する DNS ゾーンを含む IBM® DNS サービスを作成する必要もあります。DNS に IBM® CIS を使用する Power VS の標準デプロイメントとは異なり、DNS サービスには IBM® DNS を使用する必要があります。

20.6.3.1. 制限事項

IBM Power® Virtual Server 上のプライベートクラスターには、クラスターのデプロイメントに使用された既存の VPC に関連する制限のみが適用されます。

20.6.4. VPC を使用するための要件

クラスターをインストールする前に、既存の VPC およびそのサブネットを適切に設定する必要があります。このシナリオでは、インストールプログラムは VPC または VPC サブネットを作成しません。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化します。
- サブネットのルートテーブルを設定します。
- DHCP などの VPC オプションの設定

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

20.6.4.1. VPC 検証

VPC とすべてのサブネットは、既存のリソースグループ内にある必要があります。クラスターはこのリソースグループにデプロイされます。

インストールの一環として、**install-config.yaml** ファイルで以下を指定します。

- リソースグループの名前
- VPC の名前
- VPC サブネットの名前

指定したサブネットが適切であることを確認するために、インストールプログラムは、指定したすべてのサブネットが存在することを確認します。

注記

サブネット ID はサポートされていません。

20.6.4.2. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- ICMP Ingress はネットワーク全体に許可されます。
- TCP ポート 22 入力 (SSH) はネットワーク全体で許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

20.6.5. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

20.6.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

-

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

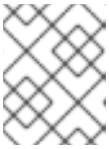
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

20.6.7. インストールプログラムの取得

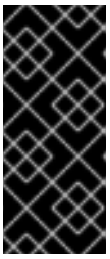
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

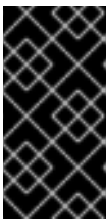
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

20.6.8. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IBMCLOUD_API_KEY=<api_key>
```

重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

20.6.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

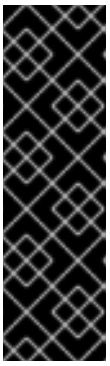
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- IBM Power® Virtual Server のインストール設定パラメーター

20.6.9.1. クラスターインストールの最小リソース要件

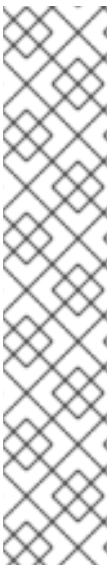
それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表20.4 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	2	16 GB	100 GB	300
コントロールプレーン	RHCOS	2	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform と Kubernetes は、ディスクのパフォーマンスの影響を受けるため、特にコントロールプレーンノードの etcd には、より高速なストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。



注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

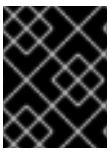
プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

20.6.9.2. IBM Power Virtual Server 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com
compute: 1 2
- architecture: ppc64le
  hyperthreading: Enabled 3
  name: worker
  platform:
    powervs:
      smtLevel: 8 4
  replicas: 3
controlPlane: 5 6
  architecture: ppc64le
  hyperthreading: Enabled 7
  name: master
  platform:
    powervs:
      smtLevel: 8 8
  replicas: 3
metadata:
  creationTimestamp: null
  name: example-private-cluster-name
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.0.0/24
  networkType: OVNKubernetes 10
  serviceNetwork:
  - 172.30.0.0/16
platform:
  powervs:
    userID: ibm-user-id
    powervsResourceGroup: "ibmcloud-resource-group"
    region: powervs-region
    vpcName: name-of-existing-vpc 11
    vpcSubnets:
    - powervs-region-example-subnet-1
    vpcRegion : vpc-region
    zone: powervs-zone
```

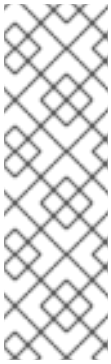
```
    serviceInstanceGUID: "powervs-region-service-instance-guid"
publish: Internal (12)
pullSecret: '{"auths": ...}' (13)
sshKey: ssh-ed25519 AAAA... (14)
```

(1) (5) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(2) (6) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。現在、両方のセクションで単一のマシンプールが定義されています。1 つのコントロールプレーンプールのみが使用されます。

(3) (7) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

(4) (8) smtLevel は、コントロールプレーンとコンピュートマシンに設定する SMT のレベルを指定します。サポートされている値は、1、2、4、8、**'off'** および **'on'** です。デフォルト値は 8 です。smtLevel を **'off'** にすると、SMT がオフに設定されます。smtlevel を **'on'** にすると、クラスターノードで SMT がデフォルト値 8 に設定されます。



注記

同時マルチスレッド (SMT) またはハイパースレッディングが有効でない場合、1 vCPU が 1 つの物理コアに相当します。有効な場合、(コアあたりのスレッド数 x ソケットあたりのコア数) x ソケット数という式で合計 vCPU が計算されます。smtLevel はコアあたりのスレッド数を制御します。SMT レベルが低い場合、クラスターノードをデプロイするときに追加の割り当てコアが必要になる場合があります。これを行うには、**install-config.yaml** ファイルの **processors** パラメーターを、OpenShift Container Platform を正常にデプロイするための要件を満たす適切な値に設定します。

(9) マシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれている必要があります。

(10) インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

(11) 既存 VPC の名前を指定します。

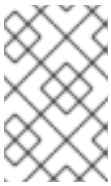
(12) クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法を指定します。プライベートクラスターをデプロイするには、公開を **internal** に設定します。

(13) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(14) クラスター内のマシンにアクセスするために使用する **sshKey** 値を指定します。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

20.6.9.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

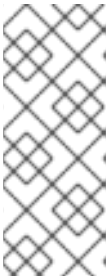
3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に .

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



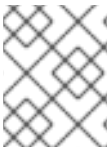
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

20.6.10. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

   ```
   apiVersion: v1
   baseDomain: cluster1.example.com
   credentialsMode: Manual (1)
   compute:
   - architecture: ppc64le
     hyperthreading: Enabled
   ```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \(1)
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \(2)
     --to=<path_to_directory_for_credentials_requests> (3)
   ```

   (1) **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。

   (2) **install-config.yaml** ファイルの場所を指定します。

   (3) **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

   このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

   サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
  --credentials-requests-dir=<path_to_credential_requests_directory> \ 1
  --name=<cluster_name> \ 2
  --output-dir=<installation_directory> \ 3
  --resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

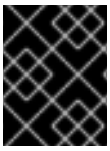
```
$ grep resourceGroup <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

20.6.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

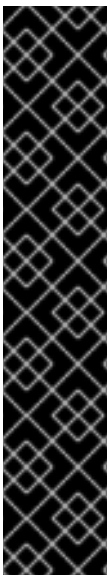
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

20.6.12. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

20.6.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

20.6.14. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

20.6.15. 次のステップ

- クラスターのカスタマイズ
- オプション: リモートヘルスレポートのオプトアウト

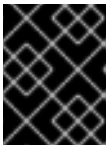
20.7. 制限されたネットワーク内の IBM POWER VIRTUAL SERVER へのクラスターのインストール

OpenShift Container Platform 4.16 では、IBM Cloud® 上の既存の Virtual Private Cloud (VPC) にインストールリリースコンテンツの内部ミラーを作成することで、制限されたネットワーク内の IBM Cloud® にクラスターをインストールできます。

20.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターをホストするように IBM Cloud® アカウントを設定 している。

- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- IBM Cloud® に既存の VPC があります。制限されたネットワークにクラスターをインストールする場合、インストーラーがプロビジョニングした VPC は使用できません。以下の要件のいずれかを満たすユーザーによってプロビジョニングされる VPC を使用する必要があります。
  - ミラーレジストリーが含まれる。
  - 別の場所でホストされるミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続がある。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。
- クラスターをインストールする前に、**ccoctl** ユーティリティーを設定している。詳細は、Cloud Credential Operator ユーティリティーの設定 を参照してください。

20.7.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

20.7.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

20.7.3. カスタム VPC の使用について

OpenShift Container Platform 4.16 では、既存の IBM® Virtual Private Cloud (VPC) のサブネットにクラスターをデプロイできます。

20.7.3.1. VPC を使用するための要件

クラスターをインストールする前に、既存の VPC およびそのサブネットを適切に設定する必要があります。このシナリオでは、インストールプログラムは VPC または VPC サブネットを作成しません。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化します。
- サブネットのルートテーブルを設定します。
- DHCP などの VPC オプションの設定

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

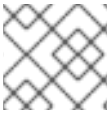
20.7.3.2. VPC 検証

VPC とすべてのサブネットは、既存のリソースグループ内にある必要があります。クラスターはこのリソースグループにデプロイされます。

インストールの一環として、**install-config.yaml** ファイルで以下を指定します。

- リソースグループの名前
- VPC の名前
- VPC サブネットの名前

指定したサブネットが適切であることを確認するために、インストールプログラムは、指定したすべてのサブネットが存在することを確認します。



注記

サブネット ID はサポートされていません。

20.7.3.3. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- ICMP Ingress はネットワーク全体に許可されます。
- TCP ポート 22 入力 (SSH) はネットワーク全体で許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

20.7.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

20.7.5. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

   a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

20.7.6. API キーのエクスポート

作成した API キーをグローバル変数として設定する必要があります。インストールプログラムは、起動時に変数を取り込み、API キーを設定します。

前提条件

- IBM Cloud® アカウント用にユーザー API キーまたはサービス ID API キーのいずれかを作成している。

手順

- アカウントの API キーをグローバル変数としてエクスポートします。

```
$ export IBMCLOUD_API_KEY=<api_key>
```

重要

変数名は指定どおりに正確に設定する必要があります。インストールプログラムは、起動時に変数名が存在することを想定しています。

20.7.7. インストール設定ファイルの作成

インストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、アクセス可能な場所にアップロードしました。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

注記

古い設定の再利用を回避するために、~/**.powervs** ディレクトリーは必ず削除してください。以下のコマンドを実行します。

```
$ rm -rf ~/.powervs
```

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットのプラットフォームとして **powervs** を選選択します。

iii. クラスターをデプロイするリージョンを選択します。

iv. クラスターをデプロイするゾーンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. 親の **platform.ibmcloud** フィールドで、クラスターをインストールする VPC のネットワークとサブネットを定義します。

```
vpcName: <existing_vpc>
vpcSubnets: <vpcSubnet>
```

platform.powervs.vpcName には、既存の IBM Cloud® の名前を指定します。**platform.powervs.vpcSubnets** には、既存のサブネットを指定します。

d. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

e. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- IBM Power® Virtual Server のインストール設定パラメーター

20.7.7.1. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表20.5 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	2	16 GB	100 GB	300

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
コントロールプレーン	RHCOS	2	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform と Kubernetes は、ディスクのパフォーマンスの影響を受けるため、特にコントロールプレーンノードの etcd には、より高速なストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

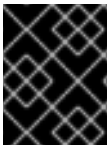
プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

20.7.7.2. IBM Power Virtual Server 用にカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    powervs:
      smtLevel: 8 (5)
  replicas: 3
compute: (6) (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    powervs:
      smtLevel: 8 (9)
    ibmcloud: {}
  replicas: 3
metadata:
  name: example-restricted-cluster-name (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (11)
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16 (12)
  networkType: OVNKubernetes (13)
  serviceNetwork:
  - 192.168.0.0/24
platform:
  powervs:
    userid: ibm-user-id
    powervsResourceGroup: "ibmcloud-resource-group" (14)
    region: "powervs-region"
    vpcRegion: "vpc-region"
    vpcName: name-of-existing-vpc (15)
    vpcSubnets: (16)
      - name-of-existing-vpc-subnet
    zone: "powervs-zone"
    serviceInstanceID: "service-instance-id"
publish: Internal
credentialsMode: Manual
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (17)
sshKey: ssh-ed25519 AAAA... (18)
additionalTrustBundle: | (19)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: (20)
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
```

```
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 10 必須。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 8 ハイパースレッディングとも呼ばれる同時マルチスレッドを有効または無効にします。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

5 9 smtLevel は、コントロールプレーンとコンピュートマシンに設定する SMT のレベルを指定します。サポートされている値は、1、2、4、8、**'off'** および **'on'** です。デフォルト値は 8 です。smtLevel を **'off'** にすると、SMT がオフに設定されます。smtlevel を **'on'** にすると、クラスターノードで SMT がデフォルト値 8 に設定されます。

注記

同時マルチスレッド (SMT) またはハイパースレッディングが有効でない場合、1 vCPU が 1 つの物理コアに相当します。有効な場合、(コアあたりのスレッド数 x ソケットあたりのコア数) x ソケット数という式で合計 vCPU が計算されます。smtLevel はコアあたりのスレッド数を制御します。SMT レベルが低い場合、クラスターノードをデプロイするときに追加の割り当てコアが必要になる場合があります。これを行うには、**install-config.yaml** ファイルの **processors** パラメーターを、OpenShift Container Platform を正常にデプロイするための要件を満たす適切な値に設定します。

11 マシン CIDR にはコンピュートマシンおよびコントロールプレーンマシンのサブネットが含まれている必要があります。

12 CIDR には、**platform.ibmcloud.controlPlaneSubnets** および **platform.ibmcloud.computeSubnets** で定義されたサブネットが含まれている必要があります。

13 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

14 既存のリソースグループの名前。既存の VPC およびサブネットはこのリソースグループにある必要があります。クラスターはこのリソースグループにデプロイされます。

15 既存 VPC の名前を指定します。

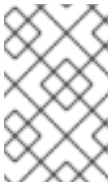
16 既存の VPC サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。リージョン内の各アベイラビリティーゾーンのサブネットを指定します。

17 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: registry.example.com または registry.example.com:5000**<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

18 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

19 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

20 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

20.7.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

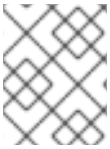
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

20.7.8. IAM を手動で作成する

クラスターをインストールするには、Cloud Credential Operator (CCO) が手動モードで動作する必要があります。インストールプログラムは CCO を手動モードに設定しますが、クラウドプロバイダーの ID とアクセス管理シークレットを指定する必要があります。

Cloud Credential Operator (CCO) ユーティリティー (**ccoctl**) を使用して、必要な IBM Cloud® リソースを作成できます。

前提条件

- **ccoctl** バイナリーを設定している。
- 既存の **install-config.yaml** ファイルがある。

手順

1. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

   ```
   apiVersion: v1
   baseDomain: cluster1.example.com
   credentialsMode: Manual (1)
   compute:
   - architecture: ppc64le
     hyperthreading: Enabled
   ```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

2. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

3. インストールプログラムが含まれているディレクトリーから、次のコマンドを実行して、インストールファイルのリリースイメージを **$RELEASE_IMAGE** 変数に設定します。

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. **--included** パラメーターには、特定のクラスター設定に必要なマニフェストのみが含まれます。
2. **install-config.yaml** ファイルの場所を指定します。
3. **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

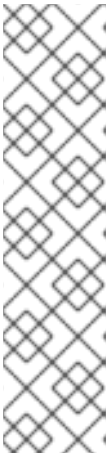
```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-ibmcos
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: IBMCloudProviderSpec
    policies:
    - attributes:
      - name: serviceName
        value: cloud-object-storage
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
      - crn:v1:bluemix:public:iam::::role:Operator
      - crn:v1:bluemix:public:iam::::role:Editor
      - crn:v1:bluemix:public:iam::::serviceRole:Reader
      - crn:v1:bluemix:public:iam::::serviceRole:Writer
    - attributes:
      - name: resourceType
        value: resource-group
      roles:
      - crn:v1:bluemix:public:iam::::role:Viewer
```

5. 各認証情報リクエストのサービス ID を作成し、定義されたポリシーを割り当て、API キーを作成し、シークレットを生成します。

```
$ ccoctl ibmcloud create-service-id \
```

```
--credentials-requests-dir=<path_to_credential_requests_directory> \ 1
--name=<cluster_name> \ 2
--output-dir=<installation_directory> \ 3
--resource-group-name=<resource_group_name> 4
```

1. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
2. OpenShift Container Platform クラスターの名前を指定します。
3. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
4. オプション: アクセスポリシーのスコープに使用されるリソースグループの名前を指定します。



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

間違ったリソースグループ名が指定された場合、ブートストラップフェーズ中にインストールが失敗します。正しいリソースグループ名を見つけるには、次のコマンドを実行します。

```
$ grep resourceGroup <installation_directory>/manifests/cluster-infrastructure-02-config.yml
```

検証

- クラスターの **manifests** ディレクトリーに適切なシークレットが生成されていることを確認してください。

20.7.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定しました。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

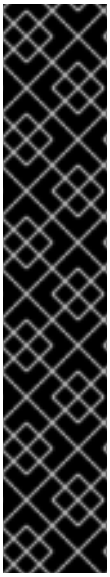
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

20.7.10. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

20.7.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- Web コンソールへのアクセス

20.7.12. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

20.7.13. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- リモートヘルスモニタリングについて

20.7.14. 次のステップ

- クラスターのカスタマイズ
- オプション: リモートヘルスレポートのオプトアウト
- オプション: 非接続クラスターの登録

20.8. IBM POWER VIRTUAL SERVER でのクラスターのアンインストール

IBM Power® Virtual Server にデプロイしたクラスターを削除できます。

20.8.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。
- **ccoctl** バイナリーを設定している。
- IBM Cloud® CLI をインストールし、VPC インフラストラクチャーサービスプラグインをインストールまたは更新している。詳細は、IBM Cloud® CLI ドキュメント の "Prerequisites" を参照してください。

手順

1. 次の条件が満たされている場合、この手順が必要です。
   - インストーラーは、インストールプロセスの一環としてリソースグループを作成しました。
   - クラスターがデプロイされた後、ユーザーまたはお使いのアプリケーションの 1 つが永続ボリューム要求 (PVC) を作成しました。

   この場合、クラスターをアンインストールするときに PVC が削除されないため、リソースグループが正常に削除されない可能性があります。失敗を防ぐには、以下を行います。

   a. CLI を使用して IBM Cloud® にログインします。

   b. PVC をリスト表示するには、次のコマンドを実行します。

   ```
   $ ibmcloud is volumes --resource-group-name <infrastructure_id>
   ```

   ボリュームのリストの詳細については、IBM Cloud® CLI のドキュメント を参照してください。

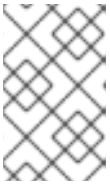
   c. PVC を削除するには、次のコマンドを実行します。

   ```
   $ ibmcloud is volume-delete --force <volume_id>
   ```

   ボリュームの削除の詳細は、IBM Cloud® CLI のドキュメント を参照してください。

2. インストールプロセスの一環として作成された API キーをエクスポートします。

```
$ export IBMCLOUD_API_KEY=<api_key>
```



注記

変数名は指定どおりに設定する必要があります。インストールプログラムは、クラスターのインストール時に作成されたサービス ID を削除するために、変数名が存在することを想定しています。

3. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

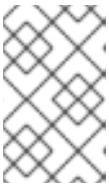
1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

- クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。
- 適切なクリーンアップを確実に行うには 、**openshift-install destroy** コマンドを最大 3 回実行する必要がある場合があります。

4. クラスター用に作成された手動の CCO クレデンシャルを削除します。

```
$ ccoctl ibmcloud delete-service-id \
    --credentials-requests-dir <path_to_credential_requests_directory> \
    --name <cluster_name>
```



注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

5. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

20.9. IBM POWER VIRTUAL SERVER のインストール設定パラメーター

IBM Power® Virtual Server に OpenShift Container Platform をデプロイする前に、クラスターとそれをホストするプラットフォームをカスタマイズするパラメーターを指定します。**install-config.yaml** ファイルを作成するときは、コマンドラインを使用して必要なパラメーターの値を指定します。その後、**install-config.yaml** ファイルを変更して、クラスターをさらにカスタマイズできます。

20.9.1. IBM Power Virtual Server で使用可能なインストール設定パラメーター

以下の表では、インストールプロセスの一部として設定できる、必須、オプション、および IBM Power Virtual Server 固有のインストール設定パラメーターを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

20.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表20.6 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object

パラメーター	説明	値
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }
platform: powervs: userID:	UserID は、ユーザーの IBM Cloud® アカウントのログインです。	文字列。例: **existing_user_id**
platform: powervs: powervsResourceGroup:	PowerVSResourceGroup は、IBM Power® Virtual Server リソースが作成されるリソースグループです。既存の VPC を使用する場合は、既存の VPC とサブネットがこのリソースグループに存在する必要があります。	文字列。例: **existing_resource_group**
platform: powervs: region:	クラスターが作成される IBM Cloud® colo リージョンを指定します。	文字列。例: **existing_region**
platform: powervs: zone:	クラスターが作成される IBM Cloud® colo リージョンを指定します。	文字列。例: **existing_zone**

20.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表20.7 ネットワークパラメーター

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 `networking:` `clusterNetwork:` `- cidr: 10.128.0.0/14` `hostPrefix: 23`
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **192.168.0.0/24**。 **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

20.9.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表20.8 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String

パラメーター	説明	値
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: smtLevel:	SMTLevel は、コントロールプレーンとコンピュートマシンに設定する SMT のレベルを指定します。有効な値は、1、2、3、4、5、6、7、8、**off**、および **on** です。	String
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。使用例: **compute.platform.powervs.sysType**。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**

パラメーター	説明	値
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**

パラメーター	説明	値
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。使用例: **controlPlane.platform.powervs.processors**。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。
platform: powervs: vpcRegion:	VPC リソースを作成する IBM Cloud® リージョンを指定します。	文字列。例: **existing_vpc_region**
platform: powervs: vpcSubnets:	クラスターリソースが作成される既存のサブネットを (名前で) 指定します。	文字列。例: **powervs_region_example_subnet**
platform: powervs: vpcName:	IBM Cloud® 名を指定します。	文字列。例: **existing_vpcName**
platform: powervs: serviceInstanceGUID:	ServiceInstanceID は、IBM Cloud® Catalog から作成された Power IAAS インスタンスの ID です。	文字列。たとえば、**existing_service_instance_GUID** です。
platform: powervs: clusterOSImage:	ClusterOSImage は、クラスターノードのデフォルトのイメージをオーバーライドする、事前に作成された IBM Power® Virtual Server のブートイメージです。	文字列。例: **existing_cluster_os_image**

パラメーター	説明	値
platform: powervs: defaultMachinePlatform:	DefaultMachinePlatform は、独自のプラットフォーム設定を定義していないマシンプールの IBM Power® Virtual Server にインストールするときに使用されるデフォルト設定です。	文字列。例: **existing_machine_platform**
platform: powervs: memoryGiB:	仮想マシンのメモリーのサイズ (GB 単位)。	有効な整数は、マシンタイプに応じて、2 GB 以上 64 GB 以下の整数である必要があります。
platform: powervs: procType:	ProcType は、インスタンスのプロセッサー共有モデルを定義します。	有効な値は、Capped、Dedicated、Shared です。
platform: powervs: processors:	Processors は、インスタンスの処理ユニットを定義します。	プロセッサーの数は 0.5 ~ 32 コアである必要があります。プロセッサーは 0.25 単位で指定する必要があります。
platform: powervs: sysType:	SysType は、インスタンスのシステムタイプを定義します。	システムタイプは **e980** または **s922** である必要があります。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

第21章 OPENSTACK へのインストール

21.1. OPENSTACK へのインストールの準備

Red Hat OpenStack Platform (RHOSP) に OpenShift Container Platform をインストールできます。

21.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。

21.1.2. OpenStack に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーによるプロビジョニングおよびユーザーによるプロビジョニングのインストールプロセスの詳細は、インストールプロセス を参照してください。

21.1.2.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる Red Hat OpenStack Platform (RHOSP) インフラストラクチャーに、クラスターをインストールできます。

- **カスタマイズによる OpenStack へのクラスターのインストール**: カスタマイズされたクラスターを RHOSP にインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の多くのカスタマイズオプションは、インストール後 に利用できます。
- **ネットワークが制限された環境での OpenStack へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを作成して、OpenShift Container Platform をネットワークが制限された環境またはネットワークの非接続環境で RHOSP にインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

21.1.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

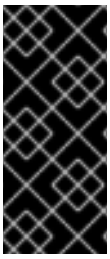
以下の方法のいずれかを使用して、独自にプロビジョニングする RHOSP インフラストラクチャーにクラスターをインストールできます。

- **独自のインフラストラクチャーでの OpenStack へのクラスターのインストール**: ユーザーによってプロビジョニングされる RHOSP インフラストラクチャーに OpenShift Container Platform をインストールできます。このインストール方法を使用して、クラスターを既存のイ

ンフラストラクチャーおよび変更と統合できます。ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの場合、Nova サーバー、Neutron ポート、セキュリティーグループなどの RHOSP リソースをすべて作成する必要があります。提供される Ansible Playbook を使用してデプロイメントプロセスを支援することができます。

21.1.3. RHOSP エンドポイントをスキャンしてレガシー HTTPS 証明書を探す

OpenShift Container Platform 4.10 以降、HTTPS 証明書にはサブジェクト代替名 (SAN) フィールドが含まれている必要があります。次のスクリプトを実行して、Red Hat Open Stack Platform (RHOSP) カタログ内の各 HTTPS エンドポイントをスキャンし、**CommonName** フィールドのみを含むレガシー証明書を探します。



重要

OpenShift Container Platform は、インストールまたは更新の前に、基盤となる RHOSP インフラストラクチャーのレガシー証明書をチェックしません。提供されているスクリプトを使用して、これらの証明書をご自身で確認してください。クラスターをインストールまたは更新する前にレガシー証明書を更新しないと、クラスターが機能しなくなります。

前提条件

- スクリプトを実行するマシンに、次のソフトウェアをインストールします。
  - Bash バージョン 4.0 以降
  - **grep**
  - OpenStack クライアント
  - **jq**
  - OpenSSL バージョン 1.1.1l 以降
- ターゲットクラウドの RHOSP クレデンシャルをマシンに入力します。

手順

1. 次のスクリプトをマシンに保存します。

```
#!/usr/bin/env bash

set -Eeuo pipefail

declare catalog san
catalog="$(mktemp)"
san="$(mktemp)"
readonly catalog san

declare invalid=0

openstack catalog list --format json --column Name --column Endpoints \
 | jq -r '.[] | .Name as $name | .Endpoints[] | select(.interface=="public") | [$name, .interface,
.url] | join(" ")' \
 | sort \
```

```
 > "$catalog"

while read -r name interface url; do
 # Ignore HTTP
 if [[ ${url#"http://"} != "$url" ]]; then
  continue
 fi

 # Remove the schema from the URL
 noschema=${url#"https://"}

 # If the schema was not HTTPS, error
 if [[ "$noschema" == "$url" ]]; then
  echo "ERROR (unknown schema): $name $interface $url"
  exit 2
 fi

 # Remove the path and only keep host and port
 noschema="${noschema%%/*}"
 host="${noschema%%:*}"
 port="${noschema##*:}"

 # Add the port if was implicit
 if [[ "$port" == "$host" ]]; then
  port='443'
 fi

 # Get the SAN fields
 openssl s_client -showcerts -servername "$host" -connect "$host:$port" </dev/null
2>/dev/null \
  | openssl x509 -noout -ext subjectAltName \
  > "$san"

 # openssl returns the empty string if no SAN is found.
 # If a SAN is found, openssl is expected to return something like:
 #
 #    X509v3 Subject Alternative Name:
 #        DNS:standalone, DNS:osp1, IP Address:192.168.2.1, IP Address:10.254.1.2
 if [[ "$(grep -c "Subject Alternative Name" "$san" || true)" -gt 0 ]]; then
  echo "PASS: $name $interface $url"
 else
  invalid=$((invalid+1))
  echo "INVALID: $name $interface $url"
 fi
done < "$catalog"

# clean up temporary files
rm "$catalog" "$san"

if [[ $invalid -gt 0 ]]; then
 echo "${invalid} legacy certificates were detected. Update your certificates to include a SAN
field."
 exit 1
else
 echo "All HTTPS certificates for this cloud are valid."
fi
```

2. スクリプトを実行します。
3. スクリプトが **INVALID** と報告する証明書を、SAN フィールドを含む証明書に置き換えます。

重要

OpenShift Container Platform 4.10 をインストールする前、またはクラスターをそのバージョンに更新する前に、すべてのレガシー HTTPS 証明書を置き換える必要があります。レガシー証明書は、次のメッセージで拒否されます。

```
x509: certificate relies on legacy Common Name field, use SANs instead
```

21.1.3.1. RHOSP エンドポイントをスキャンしてレガシー HTTPS 証明書を手動で探す

OpenShift Container Platform 4.10 以降、HTTPS 証明書にはサブジェクト代替名 (SAN) フィールドが含まれている必要があります。「レガシー HTTPS 証明書の RHOSP エンドポイントのスキャン」にリストされている前提条件ツールにアクセスできない場合は、次の手順を実行して、Red Hat OpenStack Platform (RHOSP) カタログ内の各 HTTPS エンドポイントをスキャンして、**CommonName** フィールドのみを含むレガシー証明書の Red Hat OpenStack Platform (RHOSP) カタログで各 HTTPS エンドポイントをスキャンします。

重要

OpenShift Container Platform は、インストールまたは更新の前に、基盤となる RHOSP インフラストラクチャーのレガシー証明書をチェックしません。これらの証明書を自分で確認するには、次の手順を使用します。クラスターをインストールまたは更新する前にレガシー証明書を更新しないと、クラスターが機能しなくなります。

手順

1. コマンドラインで次のコマンドを実行して、RHOSP パブリックエンドポイントの URL を表示します。

   ```
   $ openstack catalog list
   ```

   コマンドが返す各 HTTPS エンドポイントの URL を記録します。

2. 各パブリックエンドポイントについて、ホストとポートをメモします。

   ヒント

   スキーム、ポート、およびパスを削除して、エンドポイントのホストを決定します。

3. エンドポイントごとに次のコマンドを実行して、証明書の SAN フィールドを抽出します。

   a. **host** 変数を設定します。

      ```
      $ host=<host_name>
      ```

   b. **port** 変数を設定します。

      ```
      $ port=<port_number>
      ```

エンドポイントの URL にポートがない場合は、値 **443** を使用します。

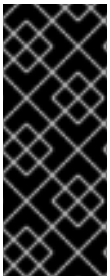
c. 証明書の SAN フィールドを取得します。

```
$ openssl s_client -showcerts -servername "$host" -connect "$host:$port" </dev/null
2>/dev/null \
    | openssl x509 -noout -ext subjectAltName
```

出力例

```
X509v3 Subject Alternative Name:
    DNS:your.host.example.net
```

各エンドポイントについて、前の例に似た出力を探します。エンドポイントの出力がない場合、そのエンドポイントの証明書は無効であるため、再発行する必要があります。



重要

OpenShift Container Platform 4.10 をインストールする前、またはクラスターをそのバージョンに更新する前に、すべてのレガシー HTTPS 証明書を置き換える必要があります。従来の証明書は拒否され、次のメッセージが表示されます。

```
x509: certificate relies on legacy Common Name field, use SANs instead
```

21.2. PREPARING TO INSTALL A CLUSTER THAT USES SR-IOV OR OVS-DPDK ON OPENSTACK

Single Root I/O Virtualization (SRIOV) または Open vSwitch を使用する OpenShift Container Platform クラスターを Red Hat OpenStack Platform (RHOSP) に Data Plane Development Kit (OVS-DPDK) とともにインストールする前に、テクノロジーごとの要件を理解し、準備タスクを実行する必要があります。

21.2.1. SR-IOV または OVS-DPDK のいずれかを使用する RHOSP 上のクラスターの要件

デプロイメントで SR-IOV または OVS-DPDK を使用する場合は、次の要件を満たす必要があります。

- RHOSP コンピュートノードは、Huge Page をサポートするフレーバーを使用する必要があります。

21.2.1.1. SR-IOV を使用する RHOSP 上のクラスターの要件

デプロイメントでシングル root I/O 仮想化 (SR-IOV) を使用するには、次の要件を満たす必要があります。

- Red Hat OpenStack Platform (RHOSP) SR-IOV デプロイメントを計画します 。
- OpenShift Container Platform は、使用する NIC をサポートする必要があります。サポートされている NIC のリストについては、『Networking』ドキュメントのHardware networksサブセクションにあるAbout Single Root I/O Virtualization (SR-IOV) hardware networksを参照してください。
- SR-IOV NIC がアタッチされるノードごとに、RHOSP クラスターに以下が必要です。

- RHOSP クォータからの 1 インスタンス
- マシンのサブネットにアタッチされた 1 つのポート
- SR-IOV 仮想機能ごとに 1 つのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

- SR-IOV デプロイメントでは、多くの場合、専用の CPU や分離された CPU などのパフォーマンスの最適化が駆使されます。パフォーマンスを最大化するには、基礎となる RHOSP デプロイメントをこれらの最適化機能を使用するように設定してから、 OpenShift Container Platform コンピュートマシンを最適化されたインフラストラクチャーで実行するように設定します。
  - パフォーマンスの良い RHOSP コンピュートノードの設定についての詳細は、パフォーマンスを向上させるためのコンピュートノードの設定 を参照してください。

21.2.1.2. OVS-DPDK を使用する RHOSP 上のクラスターの要件

デプロイメントで、Open vSwitch を Data Plane Development Kit (OVS-DPDK) とともに使用するには、以下の要件を満たす必要があります。

- 『ネットワーク機能仮想化 (NFV) のプランニングおよび設定ガイド』の OVS-DPDK デプロイメントのプランニング を参照して、Red Hat OpenStack Platform (RHOSP) OVS-DPDK デプロイメントを計画します。
- ネットワーク機能仮想化 (NFV) のプランニングおよび設定ガイドの OVS-DPDK デプロイメントの設定 に従って、RHOSP OVS-DPDK デプロイメントを設定します。

21.2.2. SR-IOV を使用するクラスターのインストールの準備

SR-IOV を使用するクラスターをインストールする前に、RHOSP を設定する必要があります。

SR-IOV を使用してクラスターをインストールする場合は、cgroup v1 を使用してクラスターをデプロイする必要があります。詳細は、Linux Control Group バージョン 1 (cgroup v1) の有効化 を参照してください。

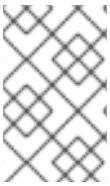
重要

cgroup v1 は非推奨の機能です。非推奨の機能は依然として OpenShift Container Platform に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

OpenShift Container Platform で非推奨となったか、削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノートの **非推奨および削除された機能**セクションを参照してください。

21.2.2.1. コンピュートマシン用の SR-IOV ネットワークの作成

Red Hat OpenStack Platform (RHOSP) デプロイメントで Single Root I/O Virtualization (SR-IOV) をサポートする場合、コンピュートマシンを実行する SR-IOV ネットワークをプロビジョニングすることができます。



注記

以下の手順では、コンピュートマシンへの接続が可能な外部のフラットネットワークおよび外部の VLAN ベースのネットワークを作成します。RHOSP のデプロイメントによっては、ネットワークの他のタイプが必要になる場合があります。

前提条件

- クラスターは SR-IOV をサポートしている。

注記

クラスターがサポートするかどうかが不明な場合は、OpenShift Container Platform SR-IOV ハードウェアネットワークについてのドキュメントを参照してください。

- RHOSP デプロイメントの一部として、無線とアップリンクのプロバイダーネットワークを作成している。これらのネットワークを表すために **radio** および **uplink** の名前がすべてのコマンド例で使用されています。

手順

1. コマンドラインで、無線の RHOSP ネットワークを作成します。

```
$ openstack network create radio --provider-physical-network radio --provider-network-type flat --external
```

2. アップリンクの RHOSP ネットワークを作成します。

```
$ openstack network create uplink --provider-physical-network uplink --provider-network-type vlan --external
```

3. 無線ネットワーク用のサブネットを作成します。

```
$ openstack subnet create --network radio --subnet-range <radio_network_subnet_range> radio
```

4. アップリンクネットワーク用のサブネットを作成します。

```
$ openstack subnet create --network uplink --subnet-range <uplink_network_subnet_range> uplink
```

21.2.3. OVS-DPDK を使用するクラスターのインストールの準備

SR-IOV を使用するクラスターをインストールする前に、RHOSP を設定する必要があります。

- RHOSP にクラスターをインストールする前に、OVS-DPDK 用のフレーバーの作成とインスタンスのデプロイ を完了します。

インストール前のタスクを実行したら、最も関連性の高い OpenShift Container Platform on RHOSP のインストール手順に従ってクラスターをインストールします。次に、このページの次のステップの下にあるタスクを実行します。

21.2.4. 次のステップ

- いずれかのデプロイメントタイプで、以下を実行します。
  - huge page をサポートする Node Tuning Operator の設定
- クラスターをデプロイした後に SR-IOV 設定を完了するには、以下を実行します。
  - SR-IOV Operator をインストール します。
  - SR-IOV ネットワークデバイスを設定 します。
  - SR-IOV コンピュートマシンを作成 します。
- パフォーマンスを向上させるためにクラスターをデプロイした後、次の参考資料を確認してください。
  - OpenStack で OVS-DPDK を使用するクラスター用のテスト Pod テンプレート
  - OpenStack で SR-IOV を使用するクラスター用のテスト Pod テンプレート
  - OpenStack で OVS-DPDK を使用するクラスター用のパフォーマンスプロファイルテンプレート

21.3. カスタマイズによる OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、Red Hat OpenStack Platform (RHOSP) にカスタマイズしたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に **install-config.yaml** でパラメーターを変更します。

21.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- OpenShift クラスターでサポートされるプラットフォーム セクションを使用して、OpenShift Container Platform 4.16 が RHOSP バージョンと互換性があることを確認した。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ブロックストレージ (Cinder) またはオブジェクトストレージ (Swift) などのストレージサービスが RHOSP にインストールされている。オブジェクトストレージは、OpenShift Container Platform レジストリークラスターデプロイメントに推奨されるストレージ技術です。詳細は、ストレージの最適化 を参照してください。
- クラスターのスケーリング、コントロールプレーンのサイジング、および etcd のパフォーマンスおよびスケーラビリティーについての理解がある。詳細は、クラスターのスケーリングに関する推奨プラクティス を参照してください。
- RHOSP でメタデータサービスが有効化されている。

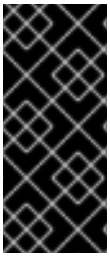
21.3.2. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表21.1 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

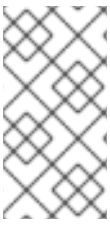
リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	88 GB
vCPU	22
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60
サーバーグループ	2 - 各マシンプールの追加のアベイラビリティーゾーンごとに 1 つ追加

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。



重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。



注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

21.3.2.1. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリーと 4 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

21.3.2.2. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリーと 2 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

21.3.2.3. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリーと 4 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

21.3.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、デフォルトの内部ロードバランシングソリューションの代わりに使用する独自の API およびアプリケーション Ingress ロードバランシングインフラストラクチャーをプロビジョニングできます。実稼働のシナリオでは、API およびアプリケーショ

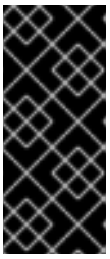
ン Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



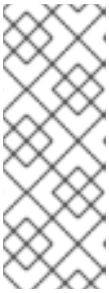
重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表21.2 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表21.3 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

注記

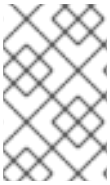
ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

21.3.2.4.1. ユーザー管理のロードバランサーを使用してデプロイされたクラスターのロードバランサー設定の例

このセクションでは、ユーザー管理のロードバランサーを使用してデプロイされたクラスターのロードバランシング要件を満たす API およびアプリケーションの ingress load balancer 設定の例を示します。

この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例21.1 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2 rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
```

```
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 5
    bind *:443
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:443 check inter 1s
    server compute1 compute1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 6
    bind *:80
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:80 check inter 1s
    server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

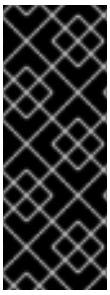
HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

21.3.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

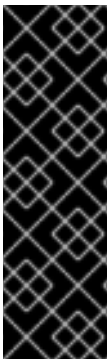


重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

21.3.4. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。



重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。

重要

RHOSP 17 では、Ceph RGW の **rgw_max_attr_size** パラメーターが 256 文字に設定されます。この設定は、コンテナーイメージを OpenShift Container Platform レジストリーにアップロードする際に問題を引き起こします。**rgw_max_attr_size** の値は、1024 文字以上に設定する必要があります。

インストールする前に、RHOSP のデプロイメントがこの問題の影響を受けるかどうか確認してください。影響を受ける場合は、Ceph RGW を再設定します。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。
- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

21.3.5. RHOSP で実行されるクラスター上のカスタムストレージを使用したイメージレジストリーの設定

Red Hat OpenStack Platform (RHOSP) にクラスターをインストールした後に、特定のアベイラビリティーゾーンにある Cinder ボリュームをレジストリーストレージとして使用できます。

手順

1. YAML ファイルを作成して、使用するストレージクラスとアベイラビリティーゾーンを指定します。以下に例を示します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: custom-csi-storageclass
provisioner: cinder.csi.openstack.org
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
parameters:
  availability: <availability_zone_name>
```

注記

OpenShift Container Platform では、選択したアベイラビリティーゾーンが存在するかどうかは確認されません。設定を適用する前に、アベイラビリティーゾーンの名前を確認してください。

2. コマンドラインから設定を適用します。

```
$ oc apply -f <storage_class_file_name>
```

出力例

```
storageclass.storage.k8s.io/custom-csi-storageclass created
```

3. ストレージクラスと **openshift-image-registry** namespace を使用する永続ボリュームクレーム (PVC) を指定する YAML ファイルを作成します。以下に例を示します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc-imageregistry
  namespace: openshift-image-registry (1)
  annotations:
    imageregistry.openshift.io: "true"
```

```
spec:
  accessModes:
  - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 100Gi (2)
  storageClassName: <your_custom_storage_class> (3)
```

(1) **openshift-image-registry** namespace を入力します。この namespace により、クラスターイメージレジストリーオペレーターは PVC を使用できます。

(2) オプション: ボリュームサイズを調整します。

(3) 作成されるストレージクラスの名前を入力します。

4. コマンドラインから設定を適用します。

```
$ oc apply -f <pvc_file_name>
```

出力例

```
persistentvolumeclaim/csi-pvc-imageregistry created
```

5. イメージレジストリー設定の元の永続ボリューム要求は、新しい要求に置き換えます。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type 'json' -p='[{"op":
"replace", "path": "/spec/storage/pvc/claim", "value": "csi-pvc-imageregistry"}]'
```

出力例

```
config.imageregistry.operator.openshift.io/cluster patched
```

数分すると、設定が更新されます。

検証

レジストリーが定義したリソースを使用していることを確認するには、以下を実行します。

1. PVC クレーム値が PVC 定義で指定した名前と同じであることを確認します。

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

出力例

```
...
status:
   ...
   managementState: Managed
   pvc:
     claim: csi-pvc-imageregistry
...
```

2. PVC のステータスが **Bound** であることを確認します。

```
$ oc get pvc -n openshift-image-registry csi-pvc-imageregistry
```

出力例

```
NAME                    STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS            AGE
csi-pvc-imageregistry   Bound    pvc-72a8f9c9-f462-11e8-b6b6-fa163e18b7b5   100Gi
RWO            custom-csi-storageclass   11m
```

21.3.6. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

外部ルータータイプのあるネットワークがネットワークリストに表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

重要

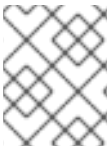
外部ネットワークの CIDR 範囲がデフォルトのネットワーク範囲のいずれかと重複している場合、インストールプロセスを開始する前に、**install-config.yaml** ファイルで一致するネットワーク範囲を変更する必要があります。

デフォルトのネットワーク範囲は以下のとおりです。

ネットワーク	範囲
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

警告

インストールプログラムにより同じ名前を持つ複数のネットワークが見つかる場合、それらのネットワークのいずれかがランダムに設定されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。



注記

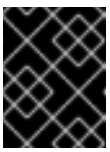
Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

21.3.7. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: <username>
      password: <password>
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: <username>
      password: <password>
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

   ```
   clouds:
     shiftstack:
       ...
       cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
   ```

   ヒント

   カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

   ```
   $ oc edit configmap -n openshift-config cloud-provider-config
   ```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

21.3.8. OpenStack Cloud Controller Manager のオプション設定

オプションで、クラスターの OpenStack Cloud Controller Manager (CCM) 設定を編集できます。この設定は、OpenShift Container Platform が Red Hat OpenStack Platform (RHOSP) と対話する方法を制御します。

設定パラメーターの完全なリストは、「OpenStack のインストール」ドキュメントの「OpenStack Cloud Controller Manager リファレンスガイド」を参照してください。

手順

1. クラスター用に生成されたマニフェストファイルがない場合は、以下のコマンドを実行して生成します。

```
$ openshift-install --dir <destination_directory> create manifests
```

2. テキストエディターで、cloud-provider 設定マニフェストファイルを開きます。以下に例を示します。

```
$ vi openshift/manifests/cloud-provider-config.yaml
```

3. CCM リファレンスガイド に従ってオプションを変更します。
負荷分散のために Octavia を設定することが一般的です。以下に例を示します。

```
#...
[LoadBalancer]
lb-provider = "amphora" 1
floating-network-id="d3deb660-4190-40a3-91f1-37326fe6ec4a" 2
create-monitor = True 3
monitor-delay = 10s 4
monitor-timeout = 10s 5
monitor-max-retries = 1 6
#...
```

1. このプロパティーは、ロードバランサーが使用する Octavia プロバイダーを設定します。**"ovn"** または **"amphora"** を値として受け入れます。OVN の使用を選択する場合は、**lb-method** を **SOURCE_IP_PORT**。

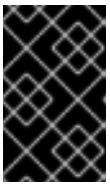
2. このプロパティーは、複数の外部ネットワークをクラスターで使用する場合に必要です。クラウドプロバイダーは、ここで指定するネットワーク上に Floating IP アドレスを作成します。

3. このプロパティーは、クラウドプロバイダーが Octavia ロードバランサーのヘルスモニターを作成するかどうかを制御します。ヘルスモニターを作成するには、値を **True** に設定します。RHOSP 16.2 の時点で、この機能は Amphora プロバイダーでのみ利用できます。

4. このプロパティーは、監視されるエンドポイントの頻度を設定します。値は **time.ParseDuration（）** 形式である必要があります。このプロパティーは、**create-monitor** プロパティーの値が **True** の場合に必要です。

5. このプロパティーは、タイムアウトする前に監視要求が開く時間を設定します。値は **time.ParseDuration（）** 形式である必要があります。このプロパティーは、**create-monitor** プロパティーの値が **True** の場合に必要です。

6. このプロパティーは、ロードバランサーがオンラインとしてマークされる前に必要なモニタリング要求の数を定義します。値は整数でなければなりません。このプロパティーは、**create-monitor** プロパティーの値が **True** の場合に必要です。



重要

変更を保存する前に、ファイルが正しく構造化されていることを確認します。プロパティーが適切なセクションに置かれていないと、クラスターが失敗することがあります。

重要

.spec.externalTrafficPolicy プロパティーの値が **Local** に設定されたサービスを使用する場合は、**create-monitor** プロパティーの値を **True** に設定する必要があります。RHOSP 16.2 の OVN Octavia プロバイダーは、ヘルスモニターをサポートしません。そのため、**lb-provider** の値が **"ovn"** に設定されている場合、**ETP** パラメーターの値が **Local** に設定されたサービスは応答しない可能性があります。

4. 変更をファイルに保存し、インストールを続行します。

ヒント

インストーラーの実行後に、クラウドプロバイダー設定を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

変更を保存した後、クラスターの再設定には多少時間がかかります。ノードが **SchedulingDisabled** のままの場合は、プロセスが完了します。

21.3.9. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

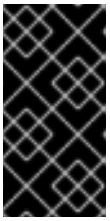
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

21.3.10. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のイン

ストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

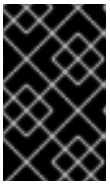
v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

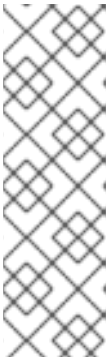
- OpenStack のインストール設定パラメーター

21.3.10.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** で

クライアントの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

21.3.10.2. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

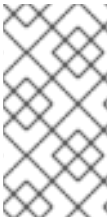
このサブネットはクラスターのプライマリーサブネットとして使用されます。デフォルトで、ノードおよびポートはこの上に作成されます。**platform.openstack.machinesSubnet** プロパティーの値をサブネットの UUID に設定すると、異なる RHOSP サブネットにノードおよびポートを作成することができます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、設定が以下の要件を満たしていることを確認してください。

- **platform.openstack.machinesSubnet** で使用されるサブネットで DHCP が有効にされている。
- **platform.openstack.machinesSubnet** の CIDR は **networking.machineNetwork** の CIDR に一致する。
- インストールプログラムのユーザーには、固定 IP アドレスを持つポートなど、このネットワークでポートを作成するパーミッションがある。

カスタムサブネットを使用するクラスターには、以下の制限があります。

- Floating IP アドレスを使用するクラスターをインストールする予定の場合には、**platform.openstack.machinesSubnet** サブネットを **externalNetwork** ネットワークに接続されているルーターに接続する必要があります。
- **platform.openstack.machinesSubnet** の値が **install-config.yaml** ファイルに設定されている場合、インストールプログラムは RHOSP マシンのプライベートネットワークまたはサブネットを作成しません。
- **platform.openstack.externalDNS** プロパティーは、カスタムサブネットと同時に使用することはできません。カスタムサブネットを使用するクラスターに DNS を追加するには、RHOSP ネットワークで DNS を設定します。



注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIPs** および **platform.openstack.ingressVIPs** の値を設定します。

重要

ネットワークの CIDR 範囲は、クラスターのインストール後に調整できません。Red Hat は、namespace ごとに作成される Pod の数を慎重に検討する必要があるため、クラスターのインストール時に範囲を決定するための直接的なガイダンスを提供していません。

21.3.10.3. ベアメタルマシンを使用したクラスターのデプロイ

クラスターでベアメタルマシンを使用する必要がある場合は、**install-config.yaml** ファイルを変更します。クラスターには、ベアメタル上でコントロールプレーンとコンピュートマシンの両方を実行させることも、コンピュートマシンのみを実行させることもできます。

注記

install-config.yaml ファイルで、ベアメタルワーカーに使用する RHOSP ネットワークが Floating IP アドレスをサポートするかどうかが反映されていることを確認します。

前提条件

- RHOSP の ベアメタルサービス (Ironic) は有効にされており、RHOSP Compute API でアクセスできる。
- ベアメタルは RHOSP フレーバー として利用可能である。
- クラスターが 16.1.6 以降、16.2.4 未満の RHOSP バージョンで実行している場合は、メタデータサービスが OpenShift Container Platform ノード上のサービスで使用できなくなる 既知の問題 により、ベアメタルワーカーは機能しません。
- RHOSP ネットワークは、仮想マシンとベアメタルサーバー接続の両方をサポートする。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。
- マシンを既存のネットワークにデプロイする必要がある場合、RHOSP サブネットがプロビジョニングされる。

- マシンをインストーラーでプロビジョニングされるネットワークに場合、RHOSP ベアメタルサービス (Ironic) はテナントネットワークで実行される Preboot eXecution Environment (PXE) ブートマシンをリッスンし、これと対話できます。
- **install-config.yaml** ファイルを OpenShift Container Platform インストールプログラムの一部として作成している。

手順

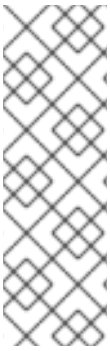
1. **install-config.yaml** ファイルで、マシンのフレーバーを編集します。
   a. ベアメタルのコントロールプレーンマシンを使用する必要がある場合は、**controlPlane.platform.openstack.type** の値をベアメタルフレーバーに変更します。
   b. **compute.platform.openstack.type** の値をベアメタルフレーバーに変更します。
   c. 既存のネットワークにマシンをデプロイする場合は、**platform.openstack.machinesSubnet** の値をネットワークの RHOSP サブネット UUID に変更します。コントロールプレーンおよびコンピュートマシンは同じサブネットを使用する必要があります。

   ベアメタルの install-config.yaml のサンプルファイル

```
controlPlane:
    platform:
      openstack:
        type: <bare_metal_control_plane_flavor> 1
...

compute:
  - architecture: amd64
    hyperthreading: Enabled
    name: worker
    platform:
      openstack:
        type: <bare_metal_compute_flavor> 2
    replicas: 3
...

platform:
    openstack:
      machinesSubnet: <subnet_UUID> 3
...
```

   1. ベアメタルのコントロールプレーンマシンを使用する必要がある場合は、この値をベアメタルのフレーバーに変更します。
   2. この値を、コンピュートマシンに使用するベアメタルのフレーバーに変更します。
   3. 既存のネットワークを使用する必要がある場合は、この値を RHOSP サブネットの UUID に変更します。

更新された **install-config.yaml** ファイルを使用してインストールプロセスを完了します。デプロイメント時に作成されるコンピュートマシンは、ファイルに追加したフレーバーを使用します。



注記

インストーラーは、ベアメタルマシンの起動中にタイムアウトする可能性があります。

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

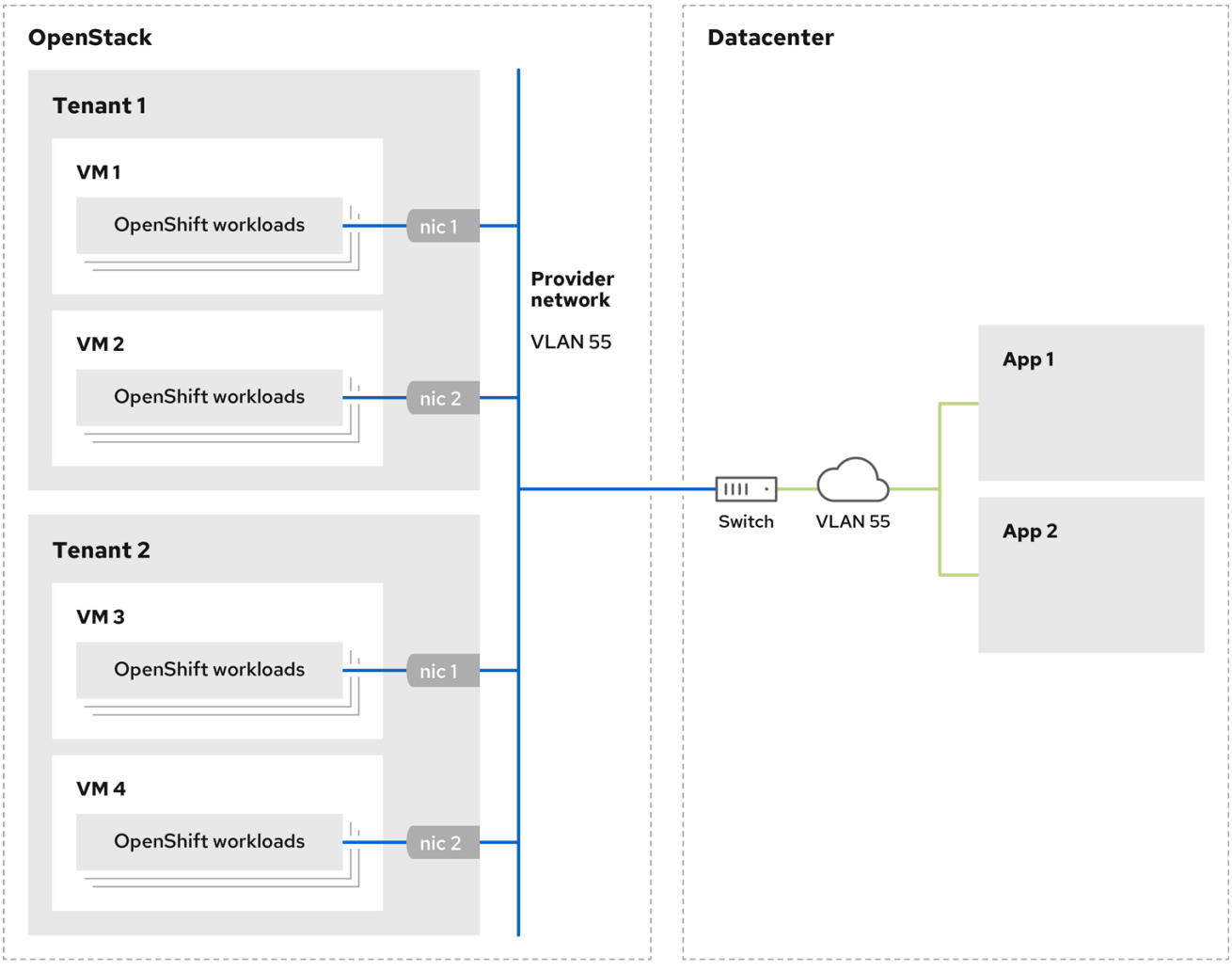
```
$ ./openshift-install wait-for install-complete --log-level debug
```

21.3.10.4. RHOSP プロバイダーネットワーク上のクラスターデプロイメント

プロバイダーネットワーク上のプライマリーネットワークインターフェイスを使用して、OpenShift Container Platform クラスターを Red Hat OpenStack Platform (RHOSP) にデプロイできます。プロバイダーネットワークは一般的に、インターネットへの到達に使用可能なパブリックネットワークに、プロジェクトが直接アクセスできるように使用します。ネットワーク作成プロセスの一環として、プロバイダーネットワークをプロジェクト間で共有することもできます。

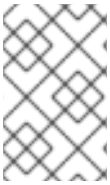
RHOSP プロバイダーネットワークは、データセンター内の既存の物理ネットワークに直接マップします。RHOSP 管理者はこれらを作成する必要があります。

以下の例では、OpenShift Container Platform ワークロードはプロバイダーネットワークを使用してデータセンターに接続されます。



プロバイダーネットワークにインストールされている OpenShift Container Platform クラスターは、テナントネットワークまたは Floating IP アドレスを必要としません。インストーラーは、インストール中にこれらのリソースを作成しません。

プロバイダーネットワークタイプの例には、フラット (タグなし) および VLAN (802.1Q タグ付き) が含まれます。



注記

クラスターは、ネットワークタイプが許可する限り多くのプロバイダーネットワーク接続をサポートできます。たとえば、VLAN ネットワークは、通常最大 4096 の接続をサポートします。

プロバイダーネットワークおよびテナントネットワークの詳細は、RHOSP のドキュメント を参照してください。

21.3.10.4.1. クラスターのインストールにおける RHOSP プロバイダーネットワーク要件

OpenShift Container Platform クラスターをインストールする前に、Red Hat OpenStack Platform (RHOSP) のデプロイメントおよびプロバイダーネットワークは、さまざまな条件を満たす必要があります。

- RHOSP ネットワークサービス (Neutron) が有効化され、RHOSP ネットワーク API 経由でアクセス可能であること。
- RHOSP ネットワークサービスでは、ポートセキュリティーと許可するアドレスペアの機能拡張が有効化 されていること。
- プロバイダーネットワークは他のテナントと共有できます。

  ヒント

  --share フラグを指定して **openstack network create** コマンドを使用して、共有できるネットワークを作成します。

- クラスターのインストールに使用する RHOSP プロジェクトは、プロバイダーネットワークと適切なサブネットを所有する必要があります。

  ヒント

  openshift という名前のプロジェクトのネットワークを作成するには、以下のコマンドを入力します。

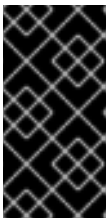
  ```
  $ openstack network create --project openshift
  ```

  openshift という名前のプロジェクトのサブネットを作成するには、以下のコマンドを入力します。

  ```
  $ openstack subnet create --project openshift
  ```

  RHOSP でのネットワークの作成に関する詳細は、プロバイダーネットワークに関するドキュメント を参照してください。

クラスターが **admin** ユーザーによって所有されている場合、そのユーザーとしてインストーラーを実行してネットワーク上でポートを作成する必要があります。



重要

プロバイダーネットワークは、クラスターの作成に使用される RHOSP プロジェクトによって所有されている必要があります。所有されていない場合は、RHOSP Compute サービス (Nova) はそのネットワークからポートを要求できません。

- プロバイダーネットワークが、デフォルトで **169.254.169.254** である RHOSP メタデータサービスの IP アドレスに到達できることを確認します。
  RHOSP SDN とネットワークサービス設定によっては、サブネットを作成する際に、ルートを提供しなければならない場合があります。以下に例を示します。

```
$ openstack subnet create --dhcp --host-route
destination=169.254.169.254/32,gateway=192.0.2.2 ...
```

- オプション: ネットワークのセキュリティーを保護するには、単一のプロジェクトへのネットワークアクセスを制限する ロールベースのアクセス制御 (RBAC) ルールを作成します。

21.3.10.4.2. プロバイダーネットワークにプライマリーインターフェイスを持つクラスターのデプロイ

Red Hat OpenStack Platform (RHOSP) プロバイダーネットワーク上にプライマリーネットワークインターフェイスを持つ OpenShift Container Platform クラスターをデプロイすることができます。

前提条件

- クラスターのインストールにおける RHOSP プロバイダーネットワーク要件に記載されているとおりに、お使いの Red Hat OpenStack Platform (RHOSP) のデプロイメントが設定されています。

手順

1. テキストエディターで **install-config.yaml** ファイルを開きます。
2. **platform.openstack.apiVIPs** プロパティーの値を API VIP の IP アドレスに設定します。
3. **platform.openstack.ingressVIPs** プロパティーの値を Ingress VIP の IP アドレスに設定します。
4. **platform.openstack.machinesSubnet** プロパティーの値をプロバイダーネットワークサブネットの UUID に設定します。
5. **networking.machineNetwork.cidr** プロパティーの値をプロバイダーネットワークサブネットの CIDR ブロックに設定します。

重要

platform.openstack.apiVIPs プロパティーおよび **platform.openstack.ingressVIPs** プロパティーはいずれも、**networking.machineNetwork.cidr** ブロックから割り当てられていない IP アドレスである必要があります。

RHOSP プロバイダーネットワークに依存するクラスターのインストール設定ファイルのセクション

```
...
platform:
  openstack:
    apiVIPs: 1
      - 192.0.2.13
    ingressVIPs: 2
      - 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

1 2 OpenShift Container Platform 4.12 以降では、**apiVIP** および **ingressVIP** 設定は非推奨です。代わりに、リスト形式を使用して、**apiVIPs** および **ingressVIPs** 設定に値を入力します。

警告

プライマリーネットワークインターフェイスにプロバイダーネットワークを使用している間は、**platform.openstack.externalNetwork** パラメーターまたは **platform.openstack.externalDNS** パラメーターを設定することはできません。

クラスターをデプロイする際に、インストーラーは **install-config.yaml** ファイルを使用してプロバイダーネットワークにクラスターをデプロイします。

ヒント

プロバイダーネットワークを含むネットワークを **platform.openstack.additionalNetworkIDs** リストに追加できます。

クラスターのデプロイ後に、Pod を追加のネットワークに接続することができます。詳細は、複数ネットワークについて を参照してください。

21.3.10.5. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル

次の **install-config.yaml** ファイルの例は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

例21.2 シングルスタックの install-config.yaml ファイルの例

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OVNKubernetes
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

例21.3 デュアルスタックの install-config.yaml ファイルの例

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd01::/48
```

```
    hostPrefix: 64
  machineNetwork:
  - cidr: 192.168.25.0/24
  - cidr: fd2e:6f44:5dd8:c956::/64
  serviceNetwork:
  - 172.30.0.0/16
  - fd02::/112
  networkType: OVNKubernetes
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiVIPs:
    - 192.168.25.10
    - fd2e:6f44:5dd8:c956:f816:3eff:fec3:5955
    ingressVIPs:
    - 192.168.25.132
    - fd2e:6f44:5dd8:c956:f816:3eff:fe40:aecb
    controlPlanePort:
      fixedIPs:
      - subnet:
          name: openshift-dual4
      - subnet:
          name: openshift-dual6
      network:
        name: openshift-dual
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

21.3.10.6. オプション: デュアルスタックネットワークを使用したクラスターの設定

RHOSP 上にデュアルスタッククラスターを作成できます。ただし、デュアルスタック設定は、IPv4 および IPv6 サブネットを持つ RHOSP ネットワークを使用している場合にのみ有効になります。

注記

RHOSP は、IPv4 シングルスタッククラスターからデュアルスタッククラスターネットワークへの変換をサポートしていません。

21.3.10.6.1. デュアルスタッククラスターの導入

手順

1. IPv4 および IPv6 サブネットを含むネットワークを作成します。**ipv6-ra-mode** および **ipv6-address-mode** フィールドで使用可能なアドレスモードは、**dhcpv6-stateful**、**dhcpv6-stateless**、および **slaac** です。

注記

デュアルスタックネットワーク MTU は、IPv6 の最小 MTU (1280) と OVN-Kubernetes カプセル化オーバーヘッド (100) の両方に対応する必要があります。

注記

DHCP はサブネット上で有効にする必要があります。

2. API ポートと Ingress VIP ポートを作成します。

3. IPv6 サブネットをルーターに追加して、ルーターのアドバタイズメントを有効にします。プロバイダーネットワークを使用している場合は、ネットワークを外部ゲートウェイとして追加することでルーターのアドバタイズメントを有効にすることができ、これにより外部接続も有効になります。

4. クラスターノードの IPv4 および IPv6 アドレスエンドポイントを設定するには、**install-config.yaml** ファイルを編集します。以下は、**install-config.yaml** ファイルの例です。

install-config.yaml の例

```
apiVersion: v1
baseDomain: mydomain.test
compute:
- name: worker
  platform:
    openstack:
      type: m1.xlarge
  replicas: 3
controlPlane:
  name: master
  platform:
    openstack:
      type: m1.xlarge
  replicas: 3
metadata:
  name: mycluster
networking:
  machineNetwork: 1
  - cidr: "192.168.25.0/24"
  - cidr: "fd2e:6f44:5dd8:c956::/64"
  clusterNetwork: 2
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd01::/48
    hostPrefix: 64
  serviceNetwork: 3
  - 172.30.0.0/16
  - fd02::/112
platform:
  openstack:
    ingressVIPs: ['192.168.25.79', 'fd2e:6f44:5dd8:c956:f816:3eff:fef1:1bad'] 4
    apiVIPs: ['192.168.25.199', 'fd2e:6f44:5dd8:c956:f816:3eff:fe78:cf36'] 5
```

```
  controlPlanePort: 6
    fixedIPs: 7
    - subnet: 8
        name: subnet-v4
        id: subnet-v4-id
    - subnet: 9
        name: subnet-v6
        id: subnet-v6-id
    network: 10
      name: dualstack
      id: network-id
```

1 2 3 IPv4 と IPv6 の両方のアドレスファミリーの IP アドレス範囲を指定する必要があります。

4 クラスターにインターフェイスを提供するための Ingress VIP サービスの仮想 IP (VIP) アドレスエンドポイントを指定します。

5 API VIP サービスの仮想 IP (VIP) アドレスエンドポイントを指定して、クラスターにインターフェイスを提供します。

6 クラスター内のすべてのノードで使用されるデュアルスタックネットワークの詳細を指定します。

7 このフィールドで指定されたサブネットの CIDR は、**network.machineNetwork** にリストされている CIDR と一致する必要があります。

8 9 **name** または **id** のいずれか、または両方の値を指定できます。

10 **ControlPlanePort** フィールドでの **network** の指定は任意です。

あるいは、IPv6 プライマリーデュアルスタッククラスターが必要な場合は、以下の例に従って **install-config.yaml** ファイルを編集します。

install-config.yaml の例

```
apiVersion: v1
baseDomain: mydomain.test
compute:
- name: worker
  platform:
    openstack:
      type: m1.xlarge
  replicas: 3
controlPlane:
  name: master
  platform:
    openstack:
      type: m1.xlarge
  replicas: 3
metadata:
  name: mycluster
networking:
  machineNetwork: 1
```

```
  - cidr: "fd2e:6f44:5dd8:c956::/64"
  - cidr: "192.168.25.0/24"
  clusterNetwork: 2
  - cidr: fd01::/48
    hostPrefix: 64
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 3
  - fd02::/112
  - 172.30.0.0/16
platform:
  openstack:
    ingressVIPs: ['fd2e:6f44:5dd8:c956:f816:3eff:fef1:1bad', '192.168.25.79'] 4
    apiVIPs: ['fd2e:6f44:5dd8:c956:f816:3eff:fe78:cf36', '192.168.25.199'] 5
    controlPlanePort: 6
      fixedIPs: 7
      - subnet: 8
          name: subnet-v6
          id: subnet-v6-id
      - subnet: 9
          name: subnet-v4
          id: subnet-v4-id
      network: 10
        name: dualstack
        id: network-id
```

1 2 3 IPv4 と IPv6 の両方のアドレスファミリーの IP アドレス範囲を指定する必要があります。

4 クラスターにインターフェイスを提供するための Ingress VIP サービスの仮想 IP (VIP) アドレスエンドポイントを指定します。

5 API VIP サービスの仮想 IP (VIP) アドレスエンドポイントを指定して、クラスターにインターフェイスを提供します。

6 クラスター内のすべてのノードで使用されるデュアルスタックネットワークの詳細を指定します。

7 このフィールドで指定されたサブネットの CIDR は、**network.machineNetwork** にリストされている CIDR と一致する必要があります。

8 9 **name** または **id** のいずれか、または両方の値を指定できます。

10 **ControlPlanePort** フィールドでの **network** の指定は任意です。

21.3.10.7. ユーザー管理のロードバランサーを使用した OpenStack 上のクラスターのインストール設定

次の **install-config.yaml** ファイルの例は、デフォルトの内部ロードバランサーではなく、ユーザー管理の外部ロードバランサーを使用するクラスターを設定する方法を示しています。

```
apiVersion: v1
baseDomain: mydomain.test
```

```
compute:
- name: worker
  platform:
    openstack:
      type: m1.xlarge
  replicas: 3
controlPlane:
  name: master
  platform:
    openstack:
      type: m1.xlarge
  replicas: 3
metadata:
  name: mycluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.10.0/24
platform:
  openstack:
    cloud: mycloud
    machinesSubnet: 8586bf1a-cc3c-4d40-bdf6-c243decc603a (1)
    apiVIPs:
    - 192.168.10.5
    ingressVIPs:
    - 192.168.10.7
    loadBalancer:
      type: UserManaged (2)
```

(1) どのロードバランサーを使用するかに関係なく、ロードバランサーはこのサブネットにデプロイされます。

(2) **UserManaged** 値は、ユーザー管理のロードバランサーを使用していることを示します。

21.3.11. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

21.3.12. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

21.3.12.1. floating IP アドレスを使用したアクセスの有効化

OpenShift Container Platform API およびクラスターアプリケーションへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

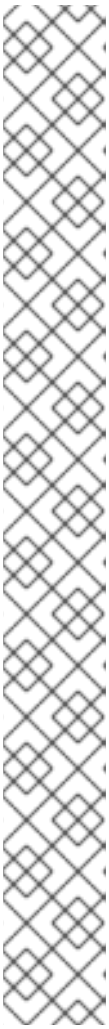
```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```



注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

4. FIP を、以下のパラメーターの値として **install-config.yaml** ファイルに追加します。
   - **platform.openstack.ingressFloatingIP**
   - **platform.openstack.apiFloatingIP**

これらの値を使用する場合には、**install-config.yaml** ファイルの **platform.openstack.externalNetwork** パラメーターの値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

21.3.12.2. Floating IP アドレスなしでのインストールの完了

Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

install-config.yaml ファイルで以下のパラメーターを定義しないでください。

- **platform.openstack.ingressFloatingIP**
- **platform.openstack.apiFloatingIP**

外部ネットワークを提供できない場合は、**platform.openstack.externalNetwork** を空白のままにすることもできます。**platform.openstack.externalNetwork** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。外部接続を独自に設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムからインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。

注記

API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

21.3.13. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

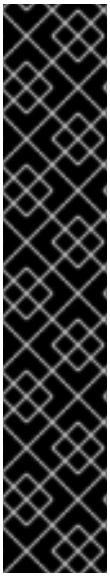
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

21.3.14. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

21.3.15. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

21.3.16. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

21.3.17. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラスタートラフィックを設定 します。
- Floating IP アドレス上でアプリケーショントラフィックを受け入れるように RHOSP を設定していない場合は、Floating IP アドレスを使用して RHOSP アクセスを設定 します。

21.4. 独自のインフラストラクチャーを使用した OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、user-provisioned infrastructure 上で実行される Red Hat OpenStack Platform (RHOSP) にクラスターをインストールできます。

独自のインフラストラクチャーを使用することで、クラスターを既存のインフラストラクチャーおよび変更と統合できます。このプロセスでは、インストーラーでプロビジョニングされるインストールの場合よりも多くの手作業が必要になります。Nova サーバー、Neutron ポート、セキュリティーグループなどのすべての RHOSP リソースを作成する必要があるためです。ただし、Red Hat では、デプロイメントプロセスを支援する Ansible Playbook を提供しています。

21.4.1. 前提条件

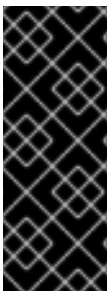
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- OpenShift クラスターでサポートされるプラットフォーム セクションを使用して、OpenShift Container Platform 4.16 が RHOSP バージョンと互換性があることを確認した。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- OpenShift Container Platform のインストール先に RHOSP アカウントがある。
- クラスターのスケーリング、コントロールプレーンのサイジング、および etcd のパフォーマンスおよびスケーラビリティーについての理解がある。詳細は、クラスターのスケーリングに関する推奨プラクティス を参照してください。
- インストールプログラムを実行するマシンには、以下が含まれる。
  - インストールプロセス時に作成したファイルを保持できる単一ディレクトリー
  - Python 3

21.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

21.4.3. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

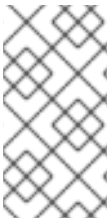
表21.4 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	88 GB
vCPU	22
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60
サーバーグループ	2 - 各マシンプールの追加のアベイラビリティーゾーンごとに 1 つ追加

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。

重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。



注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

21.4.3.1. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリーと 4 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

21.4.3.2. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリーと 2 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

21.4.3.3. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリーと 4 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

21.4.4. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでのインストールプロセスを単純化する Ansible Playbook には、複数の Python モジュールが必要です。インストーラーを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。

注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされている。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

   d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
ansible-collections-openstack
```

3. **python** コマンドが **python3** を参照していることを確認します。

```
$ sudo alternatives --set python /usr/bin/python3
```

21.4.5. インストール Playbook のダウンロード

OpenShift Container Platform を独自の Red Hat OpenStack Platform (RHOSP) インフラストラクチャーにインストールするために使用できる Ansible Playbook をダウンロードします。

前提条件

- curl コマンドラインツールがマシンで利用できる。

手順

- Playbook を作業ディレクトリーにダウンロードするには、コマンドラインから以下のスクリプトを実行します。

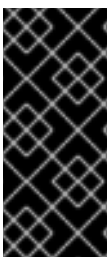
```
$ xargs -n 1 curl -O <<< '
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/common.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/down-bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/down-compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/down-control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/down-network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/down-security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/down-containers.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/inventory.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.16/upi/openstack/update-network-resources.yaml'
```

Playbook はマシンにダウンロードされます。

重要

インストールプロセス時に、Playbook を変更してデプロイメントを設定できます。

クラスターの有効期間中に、すべての Playbook を保持します。OpenShift Container Platform クラスターを RHOSP から削除するには Playbook が必要です。



重要

bootstrap.yaml、**compute-nodes.yaml**、**control-plane.yaml**、**network.yaml**、および **security-groups.yaml** ファイルに加えた編集内容は、**down-** の接頭辞が付けられた対応する Playbook に一致している必要があります。たとえば、**bootstrap.yaml** ファイルへの編集は、**down-bootstrap.yaml** ファイルにも反映される必要があります。両方のファイルを編集しない場合、サポートされるクラスターの削除プロセスは失敗します。

21.4.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

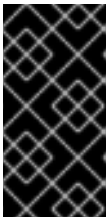
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

21.4.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

-

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

21.4.8. Red Hat Enterprise Linux CoreOS (RHCOS) イメージの作成

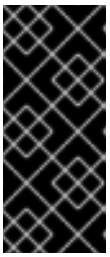
OpenShift Container Platform インストールプログラムでは、Red Hat Enterprise Linux CoreOS (RHCOS) イメージが Red Hat OpenStack Platform (RHOSP) クラスターに存在する必要があります。最新の RHCOS イメージを取得した後、RHOSP CLI を使用してこれをアップロードします。

前提条件

- RHOSP CLI がインストールされています。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、Red Hat Enterprise Linux (RHEL) 8 用の OpenShift Container Platform 4.16 の最新リリースを選択します。

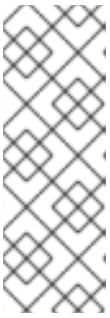


重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**をダウンロードします。

4. イメージを展開します。



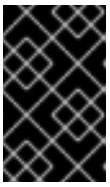
注記

クラスターが使用する前に RHOSP イメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、どのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. ダウンロードしたイメージから、RHOSP CLI を使用して **rhcos** という名前のイメージをクラスターに作成します。

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-${RHCOS_VERSION}-openstack.qcow2 rhcos
```



重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

RHOSP にイメージをアップロードした後は、インストールプログラムでイメージを利用できます。

21.4.9. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

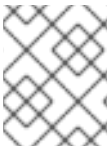
1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+---------------+-------------+
| ID                                   | Name          | Router Type |
+--------------------------------------+---------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+---------------+-------------+
```

外部ルータータイプのあるネットワークがネットワークリストに表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。



注記

Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

21.4.10. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

21.4.10.1. floating IP アドレスを使用したアクセスの有効化

OpenShift Container Platform API、クラスターアプリケーション、およびブートストラッププロセスへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. Red Hat OpenStack Platform (RHOSP) CLI を使用して、ブートストラップ FIP を作成します。

```
$ openstack floating ip create --description "bootstrap machine" <external_network>
```

4. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

5. FIP を以下の変数の値として **inventory.yaml** ファイルに追加します。
   - **os_api_fip**
   - **os_bootstrap_fip**
   - **os_ingress_fip**

これらの値を使用する場合には、**inventory.yaml** ファイルの **os_external_network** 変数の値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

21.4.10.2. Floating IP アドレスなしでのインストールの完了

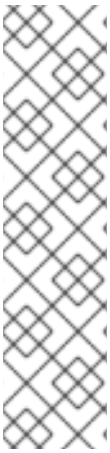
Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

inventory.yaml ファイルで、以下の変数を定義しないでください。

- **os_api_fip**
- **os_bootstrap_fip**
- **os_ingress_fip**

外部ネットワークを提供できない場合は、**os_external_network** を空白のままにすることもできます。**os_external_network** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。インストールプロセスで、ネットワークリソースを作成する際に、独自の外部接続を設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムから **wait-for** コマンドでインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。



注記

API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

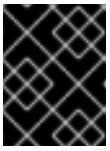
DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

21.4.11. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。



重要

パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

- RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: <username>
      password: <password>
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: <username>
      password: <password>
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

ヒント

カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: ~/**.config/openstack/clouds.yaml**)

d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
インストールプログラムはこの順序で **clouds.yaml** を検索します。

21.4.12. RHOSP でのネットワークリソースの作成

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) インストールの OpenShift Container Platform に必要なネットワークリソースを作成します。時間を節約するには、セキュリティーグループ、ネットワーク、サブネット、ルーター、およびポートを生成する指定された Ansible Playbook を実行します。

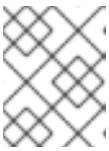
前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。

手順

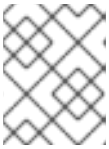
1. デュアルスタッククラスターデプロイメントの場合は、**inventory.yaml** ファイルを編集し、**os_subnet6** 属性のコメントを解除します。
2. コマンドラインで次のコマンドを実行して、ネットワークリソースを作成します。

```
$ ansible-playbook -i inventory.yaml network.yaml
```



注記

inventory.yaml Playbook の API VIP フィールドと Ingress VIP フィールドは、ネットワークポートに割り当てられた IP アドレスで上書きされます。



注記

network.yaml Playbook によって作成されたリソースは、**down-network.yaml** Playbook によって削除されます。

21.4.13. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

これで、指定したディレクトリーに **install-config.yaml** ファイルが作成されます。

関連情報

- OpenStack のインストール設定パラメーター

21.4.13.1. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

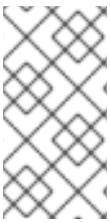
このサブネットはクラスターのプライマリーサブネットとして使用されます。デフォルトで、ノードおよびポートはこの上に作成されます。**platform.openstack.machinesSubnet** プロパティーの値をサブネットの UUID に設定すると、異なる RHOSP サブネットにノードおよびポートを作成することができます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、設定が以下の要件を満たしていることを確認してください。

- **platform.openstack.machinesSubnet** で使用されるサブネットで DHCP が有効にされている。
- **platform.openstack.machinesSubnet** の CIDR は **networking.machineNetwork** の CIDR に一致する。
- インストールプログラムのユーザーには、固定 IP アドレスを持つポートなど、このネットワークでポートを作成するパーミッションがある。

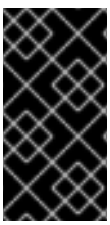
カスタムサブネットを使用するクラスターには、以下の制限があります。

- Floating IP アドレスを使用するクラスターをインストールする予定の場合には、**platform.openstack.machinesSubnet** サブネットを **externalNetwork** ネットワークに接続されているルーターに接続する必要があります。
- **platform.openstack.machinesSubnet** の値が **install-config.yaml** ファイルに設定されている場合、インストールプログラムは RHOSP マシンのプライベートネットワークまたはサブネットを作成しません。
- **platform.openstack.externalDNS** プロパティーは、カスタムサブネットと同時に使用することはできません。カスタムサブネットを使用するクラスターに DNS を追加するには、RHOSP ネットワークで DNS を設定します。



注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIPs** および **platform.openstack.ingressVIPs** の値を設定します。



重要

ネットワークの CIDR 範囲は、クラスターのインストール後に調整できません。Red Hat は、namespace ごとに作成される Pod の数を慎重に検討する必要があるため、クラスターのインストール時に範囲を決定するための直接的なガイダンスを提供していません。

21.4.13.2. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル

次の **install-config.yaml** ファイルの例は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

例21.4 シングルスタックの install-config.yaml ファイルの例

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OVNKubernetes
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

例21.5 デュアルスタックの install-config.yaml ファイルの例

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
```

```
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd01::/48
    hostPrefix: 64
  machineNetwork:
  - cidr: 192.168.25.0/24
  - cidr: fd2e:6f44:5dd8:c956::/64
  serviceNetwork:
  - 172.30.0.0/16
  - fd02::/112
  networkType: OVNKubernetes
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiVIPs:
    - 192.168.25.10
    - fd2e:6f44:5dd8:c956:f816:3eff:fec3:5955
    ingressVIPs:
    - 192.168.25.132
    - fd2e:6f44:5dd8:c956:f816:3eff:fe40:aecb
    controlPlanePort:
      fixedIPs:
      - subnet:
          name: openshift-dual4
      - subnet:
          name: openshift-dual6
      network:
        name: openshift-dual
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

21.4.13.3. マシンのカスタムサブネットの設定

インストールプログラムがデフォルトで使用する IP 範囲は、OpenShift Container Platform のインストール時に作成する Neutron サブネットと一致しない可能性があります。必要な場合は、インストール設定ファイルを編集して、新規マシンの CIDR 値を更新します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

- Python 3 がインストールされている。

手順

1. コマンドラインで、**install-config.yaml** ファイルと **inventory.yaml** ファイルが含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、次のコマンドを実行します。

```
$ python -c 'import yaml
path = "install-config.yaml"
data = yaml.safe_load(open(path))
inventory = yaml.safe_load(open("inventory.yaml"))["all"]["hosts"]["localhost"]
machine_net = [{"cidr": inventory["os_subnet_range"]}]
api_vips = [inventory["os_apiVIP"]]
ingress_vips = [inventory["os_ingressVIP"]]
ctrl_plane_port = {"network": {"name": inventory["os_network"]}, "fixedIPs": [{"subnet": {"name": inventory["os_subnet"]}}]}
if inventory.get("os_subnet6"): (1)
    machine_net.append({"cidr": inventory["os_subnet6_range"]})
    api_vips.append(inventory["os_apiVIP6"])
    ingress_vips.append(inventory["os_ingressVIP6"])
    data["networking"]["networkType"] = "OVNKubernetes"
    data["networking"]["clusterNetwork"].append({"cidr": inventory["cluster_network6_cidr"], "hostPrefix": inventory["cluster_network6_prefix"]})
    data["networking"]["serviceNetwork"].append(inventory["service_subnet6_range"])
    ctrl_plane_port["fixedIPs"].append({"subnet": {"name": inventory["os_subnet6"]}})
data["networking"]["machineNetwork"] = machine_net
data["platform"]["openstack"]["apiVIPs"] = api_vips
data["platform"]["openstack"]["ingressVIPs"] = ingress_vips
data["platform"]["openstack"]["controlPlanePort"] = ctrl_plane_port
del data["platform"]["openstack"]["externalDNS"]
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

(1) デュアルスタック (IPv4/IPv6) 環境に適用されます。

21.4.13.4. コンピュートマシンプールを空にする

独自のインフラストラクチャーを使用するインストールを実行するには、インストール設定ファイルのコンピュートマシンの数をゼロに設定します。その後、これらのマシンを手動で作成します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。

2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、手動でファイルを更新します。

   - スクリプトを使用して値を設定するには、以下を実行します。

     ```
     $ python -c '
     import yaml;
     path = "install-config.yaml";
     data = yaml.safe_load(open(path));
     data["compute"][0]["replicas"] = 0;
     open(path, "w").write(yaml.dump(data, default_flow_style=False))'
     ```

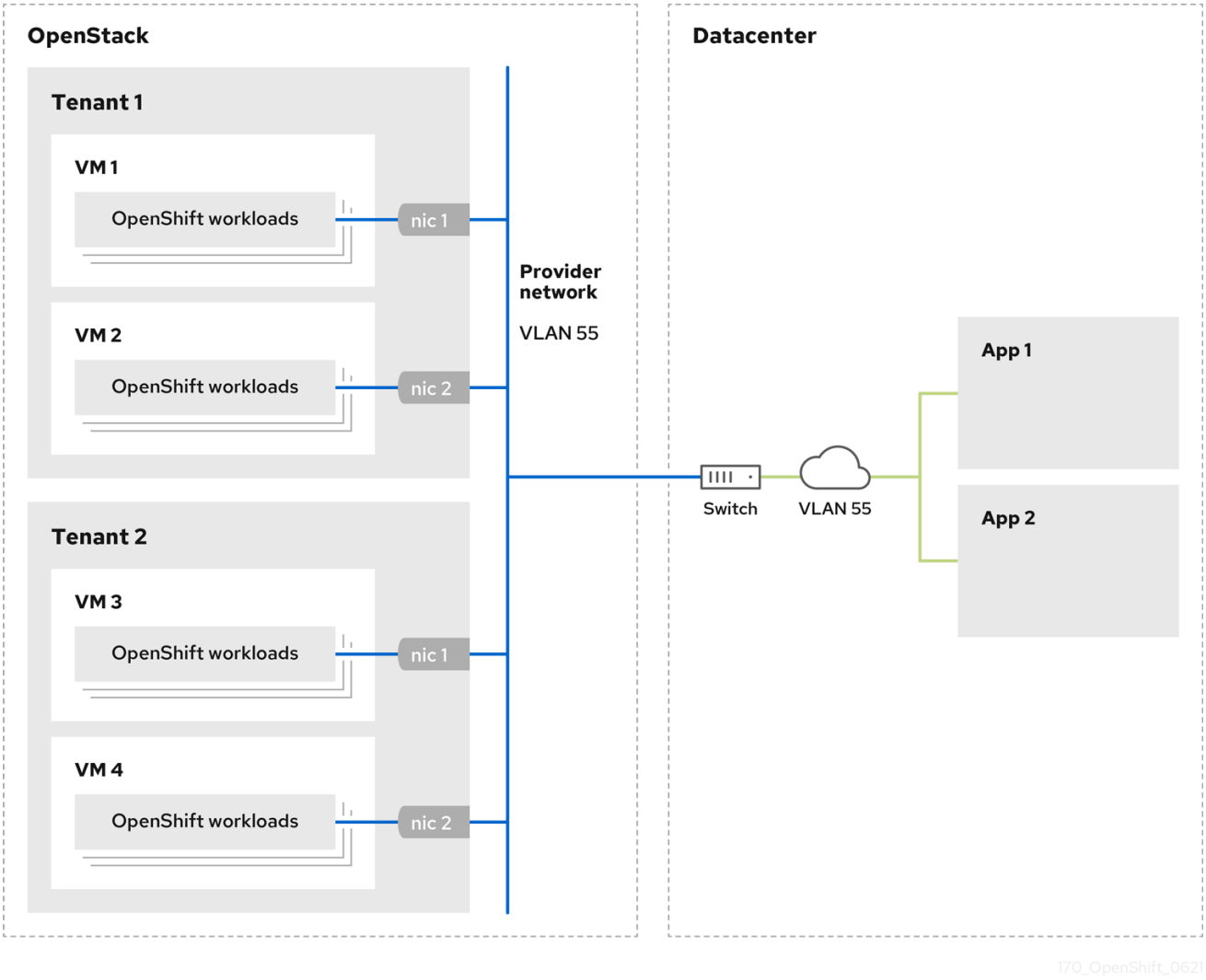
   - 値を手動で設定するには、ファイルを開き、**compute.<first entry>.replicas** の値を **0** に設定します。

21.4.13.5. RHOSP プロバイダーネットワーク上のクラスターデプロイメント

プロバイダーネットワーク上のプライマリーネットワークインターフェイスを使用して、OpenShift Container Platform クラスターを Red Hat OpenStack Platform (RHOSP) にデプロイできます。プロバイダーネットワークは一般的に、インターネットへの到達に使用可能なパブリックネットワークに、プロジェクトが直接アクセスできるように使用します。ネットワーク作成プロセスの一環として、プロバイダーネットワークをプロジェクト間で共有することもできます。

RHOSP プロバイダーネットワークは、データセンター内の既存の物理ネットワークに直接マップします。RHOSP 管理者はこれらを作成する必要があります。

以下の例では、OpenShift Container Platform ワークロードはプロバイダーネットワークを使用してデータセンターに接続されます。



プロバイダーネットワークにインストールされている OpenShift Container Platform クラスターは、テナントネットワークまたは Floating IP アドレスを必要としません。インストーラーは、インストール中にこれらのリソースを作成しません。

プロバイダーネットワークタイプの例には、フラット (タグなし) および VLAN (802.1Q タグ付き) が含まれます。

注記

クラスターは、ネットワークタイプが許可する限り多くのプロバイダーネットワーク接続をサポートできます。たとえば、VLAN ネットワークは、通常最大 4096 の接続をサポートします。

プロバイダーネットワークおよびテナントネットワークの詳細は、RHOSP のドキュメント を参照してください。

21.4.13.5.1. クラスターのインストールにおける RHOSP プロバイダーネットワーク要件

OpenShift Container Platform クラスターをインストールする前に、Red Hat OpenStack Platform (RHOSP) のデプロイメントおよびプロバイダーネットワークは、さまざまな条件を満たす必要があります。

- RHOSP ネットワークサービス (Neutron) が有効化され、RHOSP ネットワーク API 経由でアクセス可能であること。
- RHOSP ネットワークサービスでは、ポートセキュリティーと許可するアドレスペアの機能拡張が有効化 されていること。
- プロバイダーネットワークは他のテナントと共有できます。

  ヒント

  --share フラグを指定して **openstack network create** コマンドを使用して、共有できるネットワークを作成します。

- クラスターのインストールに使用する RHOSP プロジェクトは、プロバイダーネットワークと適切なサブネットを所有する必要があります。

  ヒント

  openshift という名前のプロジェクトのネットワークを作成するには、以下のコマンドを入力します。

  ```
  $ openstack network create --project openshift
  ```

  openshift という名前のプロジェクトのサブネットを作成するには、以下のコマンドを入力します。

  ```
  $ openstack subnet create --project openshift
  ```

  RHOSP でのネットワークの作成に関する詳細は、プロバイダーネットワークに関するドキュメント を参照してください。

  クラスターが **admin** ユーザーによって所有されている場合、そのユーザーとしてインストーラーを実行してネットワーク上でポートを作成する必要があります。

  重要

  プロバイダーネットワークは、クラスターの作成に使用される RHOSP プロジェクトによって所有されている必要があります。所有されていない場合は、RHOSP Compute サービス (Nova) はそのネットワークからポートを要求できません。

- プロバイダーネットワークが、デフォルトで **169.254.169.254** である RHOSP メタデータサービスの IP アドレスに到達できることを確認します。
  RHOSP SDN とネットワークサービス設定によっては、サブネットを作成する際に、ルートを提供しなければならない場合があります。以下に例を示します。

  ```
  $ openstack subnet create --dhcp --host-route
  destination=169.254.169.254/32,gateway=192.0.2.2 ...
  ```

- オプション: ネットワークのセキュリティーを保護するには、単一のプロジェクトへのネットワークアクセスを制限する ロールベースのアクセス制御 (RBAC) ルールを作成します。

21.4.13.5.2. プロバイダーネットワークにプライマリーインターフェイスを持つクラスターのデプロイ

Red Hat OpenStack Platform (RHOSP) プロバイダーネットワーク上にプライマリーネットワークインターフェイスを持つ OpenShift Container Platform クラスターをデプロイすることができます。

前提条件

- クラスターのインストールにおける RHOSP プロバイダーネットワーク要件に記載されているとおりに、お使いの Red Hat OpenStack Platform (RHOSP) のデプロイメントが設定されています。

手順

1. テキストエディターで **install-config.yaml** ファイルを開きます。
2. **platform.openstack.apiVIPs** プロパティーの値を API VIP の IP アドレスに設定します。
3. **platform.openstack.ingressVIPs** プロパティーの値を Ingress VIP の IP アドレスに設定します。
4. **platform.openstack.machinesSubnet** プロパティーの値をプロバイダーネットワークサブネットの UUID に設定します。
5. **networking.machineNetwork.cidr** プロパティーの値をプロバイダーネットワークサブネットの CIDR ブロックに設定します。

重要

platform.openstack.apiVIPs プロパティーおよび **platform.openstack.ingressVIPs** プロパティーはいずれも、**networking.machineNetwork.cidr** ブロックから割り当てられていない IP アドレスである必要があります。

RHOSP プロバイダーネットワークに依存するクラスターのインストール設定ファイルのセクション

```
...
platform:
  openstack:
    apiVIPs: 1
      - 192.0.2.13
    ingressVIPs: 2
      - 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

1 2 OpenShift Container Platform 4.12 以降では、**apiVIP** および **ingressVIP** 設定は非推奨です。代わりに、リスト形式を使用して、**apiVIPs** および **ingressVIPs** 設定に値を入力します。

警告

プライマリーネットワークインターフェイスにプロバイダーネットワークを使用している間は、**platform.openstack.externalNetwork** パラメーターまたは **platform.openstack.externalDNS** パラメーターを設定することはできません。

クラスターをデプロイする際に、インストーラーは **install-config.yaml** ファイルを使用してプロバイダーネットワークにクラスターをデプロイします。

ヒント

プロバイダーネットワークを含むネットワークを **platform.openstack.additionalNetworkIDs** リストに追加できます。

クラスターのデプロイ後に、Pod を追加のネットワークに接続することができます。詳細は、複数ネットワークについて を参照してください。

21.4.14. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシン、コンピュートマシンセット、およびコントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- コンピュートマシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5. メタデータファイルの **infraID** キーを環境変数としてエクスポートします。

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

ヒント

metadata.json から **infraID** キーを抽出し、作成するすべての RHOSP リソースの接頭辞として使用します。これを実行することで、同じプロジェクトで複数のデプロイメントを実行する際に名前の競合が発生しないようにします。

21.4.15. ブートストラップ Ignition ファイルの準備

OpenShift Container Platform インストールプロセスは、ブートストラップ Ignition 設定ファイルから作成されるブートストラップマシンに依存します。

ファイルを編集し、アップロードします。次に、Red Hat OpenStack Platform (RHOSP) がプライマリーファイルをダウンロードする際に使用するセカンダリーブートストラップ Ignition 設定ファイルを作成します。

前提条件

- インストーラープログラムが生成するブートストラップ Ignition ファイル **bootstrap.ign** があります。
- インストーラーのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、**Kubernetes マニフェストおよび Ignition 設定ファイルの作成** を参照してください。
- HTTP(S) でアクセス可能な方法でブートストラップ Ignition ファイルを保存できます。
  - 記載された手順では RHOSP イメージサービス (Glance) を使用しますが、RHOSP ストレージサービス (Swift)、Amazon S3、内部 HTTP サーバー、またはアドホックの Nova サーバーを使用することもできます。

手順

1. 以下の Python スクリプトを実行します。スクリプトはブートストラップ Ignition ファイルを変更して、ホスト名および利用可能な場合は、実行時の CA 証明書ファイルを設定します。

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
```

```
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64
    }
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64
        }
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

2. RHOSP CLI を使用して、ブートストラップ Ignition ファイルを使用するイメージを作成します。

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

3. イメージの詳細を取得します。

```
$ openstack image show <image_name>
```

file 値をメモします。これは **v2/images/<image_ID>/file** パターンをベースとしています。

注記

作成したイメージがアクティブであることを確認します。

4. イメージサービスのパブリックアドレスを取得します。

```
$ openstack catalog show image
```

5. パブリックアドレスとイメージ **file** 値を組み合わせ、結果を保存場所として保存します。この場所は、**<image_service_public_URL>/v2/images/<image_ID>/file** パターンをベースとしています。

6. 認証トークンを生成し、トークン ID を保存します。

```
$ openstack token issue -c id -f value
```

7. **$INFRA_ID-bootstrap-ignition.json** というファイルに以下のコンテンツを挿入し、独自の値に一致するようにプレースホルダーを編集します。

```
{
  "ignition": {
    "config": {
      "merge": [{
        "source": "<storage_url>", 1
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>" 4
        }]
      }
    },
    "version": "3.2.0"
  }
}
```

1. **ignition.config.append.source** の値をブートストラップ Ignition ファイルのストレージ URL に置き換えます。
2. **httpHeaders** の **name** を **"X-Auth-Token"** に設定します。
3. **httpHeaders** の **value** をトークンの ID に設定します。
4. ブートストラップ Ignition ファイルサーバーが自己署名証明書を使用する場合は、base64 でエンコードされた証明書を含めます。

8. セカンダリー Ignition 設定ファイルを保存します。

ブートストラップ Ignition データはインストール時に RHOSP に渡されます。

警告

ブートストラップ Ignition ファイルには、**clouds.yaml** 認証情報などの機密情報が含まれます。これを安全な場所に保存し、インストールプロセスの完了後に削除します。

21.4.16. RHOSP でのコントロールプレーンの Ignition 設定ファイルの作成

独自のインフラストラクチャーを使用して OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールするには、コントロールプレーンの Ignition 設定ファイルが必要です。複数の設定ファイルを作成する必要があります。

注記

ブートストラップ Ignition 設定と同様に、各コントロールプレーンマシンのホスト名を明示的に定義する必要があります。

前提条件

- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、Kubernetes マニフェストおよび Ignition 設定ファイルの作成を参照してください。

手順

- コマンドラインで、以下の Python スクリプトを実行します。

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
storage = ignition.get('storage', {});
files = storage.get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
storage['files'] = files;
ignition['storage'] = storage
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

以下の 3 つのコントロールプレーン Ignition ファイルが作成されます。**<INFRA_ID>-master-0-ignition.json**、**<INFRA_ID>-master-1-ignition.json**、および **<INFRA_ID>-master-2-ignition.json**。

21.4.17. RHOSP でのネットワークリソースの更新

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) インストールの OpenShift Container Platform に必要なネットワークリソースを更新します。

前提条件

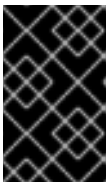
- Python 3 がマシンにインストールされている。
- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。

手順

1. オプション: 外部ネットワークの値を **inventory.yaml** Playbook に追加します。

inventory.yaml Ansible Playbook の外部ネットワーク値の例

```
...
    # The public network providing connectivity to the cluster. If not
    # provided, the cluster external connectivity must be provided in another
    # way.

    # Required for os_api_fip, os_ingress_fip, os_bootstrap_fip.
    os_external_network: 'external'
...
```



重要

inventory.yaml ファイルの **os_external_network** の値を指定しなかった場合は、仮想マシンが Glance および外部接続にアクセスできるようにする必要があります。

2. オプション: 外部ネットワークおよび Floating IP (FIP) アドレスの値を **inventory.yaml** Playbook に追加します。

inventory.yaml Ansible Playbook の FIP 値の例

```
...
    # OpenShift API floating IP address. If this value is non-empty, the
    # corresponding floating IP will be attached to the Control Plane to
    # serve the OpenShift API.
    os_api_fip: '203.0.113.23'

    # OpenShift Ingress floating IP address. If this value is non-empty, the
    # corresponding floating IP will be attached to the worker nodes to serve
    # the applications.
    os_ingress_fip: '203.0.113.19'

    # If this value is non-empty, the corresponding floating IP will be
    # attached to the bootstrap machine. This is needed for collecting logs
    # in case of install failure.
    os_bootstrap_fip: '203.0.113.20'
```

重要

os_api_fip および **os_ingress_fip** の値を定義しない場合、インストール後のネットワーク設定を実行する必要があります。

os_bootstrap_fip の値を定義しなかった場合、インストールプログラムが失敗したインストールからデバッグ情報をダウンロードできません。

詳細は、「環境へのアクセスの有効化」を参照してください。

3. コマンドラインで、**security-groups.yaml** Playbook を実行してセキュリティーグループを作成します。

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

4. コマンドラインで、**update-network-resources.yaml** Playbook を実行してネットワークリソースを更新します。

```
$ ansible-playbook -i inventory.yaml update-network-resources.yaml 1
```

1 この Playbook は、ネットワーク、サブネット、ポート、ルーターにタグを追加します。また、Floating IP アドレスを API ポートと Ingress ポートに接続し、それらのポートのセキュリティーグループを設定します。

5. オプション: Nova サーバーが使用するデフォルトのリゾルバーを制御する必要がある場合は、RHOSP CLI コマンドを実行します。

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

6. オプション: 作成した **inventory.yaml** ファイルを使用して、インストールをカスタマイズできます。たとえば、ベアメタルマシンを使用するクラスターをデプロイすることができます。

21.4.17.1. ベアメタルマシンを使用したクラスターのデプロイ

クラスターがベアメタルマシンを使用する必要がある場合は、**inventory.yaml** ファイルを変更します。クラスターには、ベアメタル上でコントロールプレーンとコンピュートマシンの両方を実行させることも、コンピュートマシンのみを実行させることもできます。

注記

install-config.yaml ファイルで、ベアメタルワーカーに使用する RHOSP ネットワークが Floating IP アドレスをサポートするかどうかが反映されていることを確認します。

前提条件

- RHOSP の ベアメタルサービス (Ironic) は有効にされており、RHOSP Compute API でアクセスできる。
- ベアメタルは RHOSP フレーバー として利用可能である。
- クラスターが 16.1.6 以降、16.2.4 未満の RHOSP バージョンで実行している場合は、メタデータサービスが OpenShift Container Platform ノード上のサービスで使用できなくなる 既知の問題 により、ベアメタルワーカーは機能しません。
- RHOSP ネットワークは、仮想マシンとベアメタルサーバー接続の両方をサポートする。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。
- マシンを既存のネットワークにデプロイする必要がある場合、RHOSP サブネットがプロビジョニングされる。
- マシンをインストーラーでプロビジョニングされるネットワークに場合、RHOSP ベアメタルサービス (Ironic) はテナントネットワークで実行される Preboot eXecution Environment (PXE) ブートマシンをリッスンし、これと対話できます。

- **inventory.yaml** ファイルを OpenShift Container Platform インストールプロセスの一部として作成している。

手順

1. **inventory.yaml** ファイルで、マシンのフレーバーを編集します。
   a. ベアメタルコントロールプレーンマシンを使用する場合は、**os_flavor_master** の値をベアメタルフレーバーに変更します。
   b. **os_flavor_worker** の値をベアメタルフレーバーに変更します。

ベアメタルの inventory.yaml のサンプルファイル

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'my-bare-metal-flavor' 1
      os_flavor_worker: 'my-bare-metal-flavor' 2
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
...
```

1. ベアメタルのコントロールプレーンマシンを使用する必要がある場合は、この値をベアメタルのフレーバーに変更します。
2. この値を、コンピュートマシンに使用するベアメタルのフレーバーに変更します。

更新された **inventory.yaml** ファイルを使用してインストールプロセスを完了します。デプロイメント時に作成されるマシンは、ファイルに追加したフレーバーを使用します。

注記

インストーラーは、ベアメタルマシンの起動中にタイムアウトする可能性があります。

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

21.4.18. RHOSP でのブートストラップマシンの作成

ブートストラップマシンを作成し、これに Red Hat OpenStack Platform (RHOSP) で実行するために必要なネットワークアクセスを付与します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、 **common.yaml**、および **bootstrap.yaml** Ansible Playbook は共通ディレクトリーにある。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで、**bootstrap.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

3. ブートストラップサーバーがアクティブになった後に、ログを表示し、Ignition ファイルが受信されたことを確認します。

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

21.4.19. RHOSP でのコントロールプレーンの作成

生成した Ignition 設定ファイルを使用して 3 つのコントロールプレーンマシンを作成します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
- **inventory.yaml**、 **common.yaml**、および **control-plane.yaml** Ansible Playbook は共通ディレクトリーにあります。
- コントロールプレーンの Ignition 設定ファイルの作成で作成された 3 つの Ignition ファイルがある。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コントロールプレーンの Ignition 設定ファイルが作業ディレクトリーにまだない場合は、ファイルを作業ディレクトリーにコピーします。
3. コマンドラインで、**control-plane.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

4. 以下のコマンドを実行してブートストラッププロセスをモニターします。

```
$ openshift-install wait-for bootstrap-complete
```

コントロールプレーンマシンが実行され、クラスターに参加していることを確認できるメッセージが表示されます。

```
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

21.4.20. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

21.4.21. RHOSP からのブートストラップリソースの削除

不要になったブートストラップリソースを削除します。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。

- **inventory.yaml**、**common.yaml**、および **down-bootstrap.yaml** Ansible Playbook が共通ディレクトリーにある。
- コントロールプレーンマシンが実行中である。
  - マシンのステータスが不明な場合は、クラスターステータスの確認を参照してください。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで、**down-bootstrap.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

ブートストラップポート、サーバー、および Floating IP アドレスが削除されます。

警告

ブートストラップ Ignition ファイル URL をまだ無効にしていない場合は、無効にしてください。

21.4.22. RHOSP でのコンピュートマシンの作成

コントロールプレーンの起動後、コンピュートマシンを作成します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、 **common.yaml**、および **compute-nodes.yaml** Ansible Playbook が共通ディレクトリーにある。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。
- コントロールプレーンが有効である。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

次のステップ

- マシンの証明書署名要求を承認します。

21.4.23. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
```

```
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

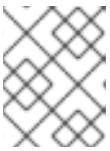
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.29.4
   master-1  Ready     master  73m  v1.29.4
   master-2  Ready     master  74m  v1.29.4
   worker-0  Ready     worker  11m  v1.29.4
   worker-1  Ready     worker  11m  v1.29.4
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

21.4.24. インストールの正常な実行の確認

OpenShift Container Platform のインストールが完了していることを確認します。

前提条件

- インストールプログラム (**openshift-install**) があります。

手順

- コマンドラインで、以下を入力します。

```
$ openshift-install --log-level debug wait-for install-complete
```

プログラムはコンソール URL と管理者のログイン情報を出力します。

21.4.25. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

21.4.26. 次のステップ

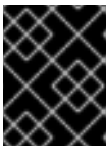
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラスタートラフィックを設定 します。
- Floating IP アドレス上でアプリケーショントラフィックを受け入れるように RHOSP を設定していない場合は、Floating IP アドレスを使用して RHOSP アクセスを設定 します。

21.5. ネットワークが制限された環境での OPENSTACK へのクラスターのインストール

OpenShift Container Platform 4.16 では、インストールリリースコンテンツの内部ミラーを作成して、制限されたネットワーク内で Red Hat OpenStack Platform (RHOSP) にクラスターをインストールできます。

21.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- OpenShift クラスターでサポートされるプラットフォーム セクションを使用して、OpenShift Container Platform 4.16 が RHOSP バージョンと互換性があることを確認した。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターのスケーリング、コントロールプレーンのサイジング、および etcd のパフォーマンスおよびスケーラビリティーについての理解がある。詳細は、クラスターのスケーリングに関する推奨プラクティス を参照してください。
- RHOSP でメタデータサービスが有効化されている。

21.5.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

21.5.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

21.5.3. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表21.5 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3
ポート	15

リソース	値
ルーター	1
サブネット	1
RAM	88 GB
vCPU	22
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60
サーバーグループ	2 - 各マシンプールの追加のアベイラビリティーゾーンごとに 1 つ追加

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。

重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。

注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

21.5.3.1. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス

- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリーと 4 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

21.5.3.2. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリーと 2 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

21.5.3.3. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリーと 4 つの vCPU を備えたフレーバー
- RHOSP クォータから少なくとも 100 GB のストレージ容量

21.5.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。

- クラスターの更新を実行するために必要なパッケージを取得します。

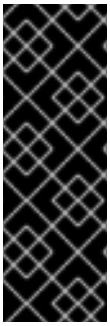
21.5.5. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。

重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。



重要

RHOSP 17 では、Ceph RGW の **rgw_max_attr_size** パラメーターが 256 文字に設定されます。この設定は、コンテナーイメージを OpenShift Container Platform レジストリーにアップロードする際に問題を引き起こします。**rgw_max_attr_size** の値は、1024 文字以上に設定する必要があります。

インストールする前に、RHOSP のデプロイメントがこの問題の影響を受けるかどうか確認してください。影響を受ける場合は、Ceph RGW を再設定します。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。
- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

21.5.6. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。

   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: <username>
           password: <password>
           user_domain_name: Default
           project_domain_name: Default
       dev-env:
         region_name: RegionOne
         auth:
           username: <username>
           password: <password>
           project_name: 'devonly'
           auth_url: 'https://10.10.14.22:5001/v2.0'
     ```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

      ```
      clouds:
        shiftstack:
          ...
          cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
      ```

      ヒント

      カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

      ```
      $ oc edit configmap -n openshift-config cloud-provider-config
      ```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

b. 現行ディレクトリー

c. Unix 固有のユーザー設定ディレクトリー (例: ~/**.config/openstack/clouds.yaml**)

d. Unix 固有のサイト設定ディレクトリー (例: /**etc/openstack/clouds.yaml**)
インストールプログラムはこの順序で **clouds.yaml** を検索します。

21.5.7. OpenStack Cloud Controller Manager のオプション設定

オプションで、クラスターの OpenStack Cloud Controller Manager (CCM) 設定を編集できます。この設定は、OpenShift Container Platform が Red Hat OpenStack Platform (RHOSP) と対話する方法を制御します。

設定パラメーターの完全なリストは、「OpenStack のインストール」ドキュメントの「OpenStack Cloud Controller Manager リファレンスガイド」を参照してください。

手順

1. クラスター用に生成されたマニフェストファイルがない場合は、以下のコマンドを実行して生成します。

```
$ openshift-install --dir <destination_directory> create manifests
```

2. テキストエディターで、cloud-provider 設定マニフェストファイルを開きます。以下に例を示します。

```
$ vi openshift/manifests/cloud-provider-config.yaml
```

3. CCM リファレンスガイド に従ってオプションを変更します。
負荷分散のために Octavia を設定することが一般的です。以下に例を示します。

```
#...
[LoadBalancer]
lb-provider = "amphora" (1)
floating-network-id="d3deb660-4190-40a3-91f1-37326fe6ec4a" (2)
create-monitor = True (3)
monitor-delay = 10s (4)
monitor-timeout = 10s (5)
monitor-max-retries = 1 (6)
#...
```

(1) このプロパティーは、ロードバランサーが使用する Octavia プロバイダーを設定します。**"ovn"** または **"amphora"** を値として受け入れます。OVN の使用を選択する場合は、**lb-method** を **SOURCE_IP_PORT**。

(2) このプロパティーは、複数の外部ネットワークをクラスターで使用する場合に必要です。クラウドプロバイダーは、ここで指定するネットワーク上に Floating IP アドレスを作成します。

(3) このプロパティーは、クラウドプロバイダーが Octavia ロードバランサーのヘルスモニターを作成するかどうかを制御します。ヘルスモニターを作成するには、値を **True** に設定します。RHOSP 16.2 の時点で、この機能は Amphora プロバイダーでのみ利用できま

す。

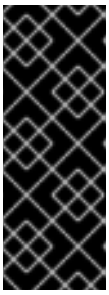
4 このプロパティーは、監視されるエンドポイントの頻度を設定します。値は **time.ParseDuration（）** 形式である必要があります。このプロパティーは、**create-monitor** プロパティーの値が **True** の場合に必要です。

5 このプロパティーは、タイムアウトする前に監視要求が開く時間を設定します。値は **time.ParseDuration（）** 形式である必要があります。このプロパティーは、**create-monitor** プロパティーの値が **True** の場合に必要です。

6 このプロパティーは、ロードバランサーがオンラインとしてマークされる前に必要なモニタリング要求の数を定義します。値は整数でなければなりません。このプロパティーは、**create-monitor** プロパティーの値が **True** の場合に必要です。

重要

変更を保存する前に、ファイルが正しく構造化されていることを確認します。プロパティーが適切なセクションに置かれていないと、クラスターが失敗することがあります。



重要

.spec.externalTrafficPolicy プロパティーの値が **Local** に設定されたサービスを使用する場合は、**create-monitor** プロパティーの値を **True** に設定する必要があります。RHOSP 16.2 の OVN Octavia プロバイダーは、ヘルスモニターをサポートしません。そのため、**lb-provider** の値が **"ovn"** に設定されている場合、**ETP** パラメーターの値が **Local** に設定されたサービスは応答しない可能性があります。

4. 変更をファイルに保存し、インストールを続行します。

ヒント

インストーラーの実行後に、クラウドプロバイダー設定を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

変更を保存した後、クラスターの再設定には多少時間がかかります。ノードが **SchedulingDisabled** のままの場合は、プロセスが完了します。

21.5.8. ネットワークが制限されたインストール用の RHCOS イメージの作成

Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードし、OpenShift Container Platform をネットワークが制限された Red Hat OpenStack Platform (RHOSP) 環境にインストールします。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、プログラムはミラーレジストリースト上に置かれます。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、RHEL 8 用の OpenShift Container Platform 4.16 の最新リリースを選択します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**イメージをダウンロードします。
4. イメージを展開します。

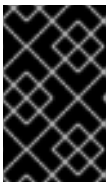
注記

クラスターが使用する前にイメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、どのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. 圧縮解除したイメージを、Glance などの bastion サーバーからアクセス可能な場所にアップロードします。以下に例を示します。

```
$ openstack image create --file rhcos-44.81.202003110027-0-openstack.x86_64.qcow2 --
disk-format qcow2 rhcos-${RHCOS_VERSION}
```



重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

これで、イメージが制限されたインストールで利用可能になります。OpenShift Container Platform デプロイメントで使用するイメージの名前または場所をメモします。

21.5.9. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、アクセス可能な場所にアップロードしました。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

2. **install-config.yaml** ファイルで、**platform.openstack.clusterOSImage** の値をイメージの場所または名前に設定します。以下に例を示します。

```
platform:
  openstack:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
openstack.x86_64.qcow2.gz?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、**<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

c. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

d. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。
パラメーターの詳細については、インストール設定パラメーターを参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- OpenStack のインストール設定パラメーター

21.5.9.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

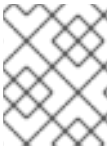
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

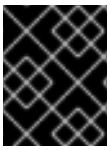


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

21.5.9.2. 制限された OpenStack インストールのカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。



重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OVNKubernetes
platform:
  openstack:
    region: region1
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
additionalTrustBundle: |
```

```
    -----BEGIN CERTIFICATE-----

    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

    -----END CERTIFICATE-----

imageContentSources:
- mirrors:
  - <mirror_registry>/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_registry>/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

21.5.10. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

21.5.11. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

21.5.11.1. floating IP アドレスを使用したアクセスの有効化

OpenShift Container Platform API およびクラスターアプリケーションへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

4. FIP を、以下のパラメーターの値として **install-config.yaml** ファイルに追加します。
    - **platform.openstack.ingressFloatingIP**
    - **platform.openstack.apiFloatingIP**

これらの値を使用する場合には、**install-config.yaml** ファイルの **platform.openstack.externalNetwork** パラメーターの値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

21.5.11.2. Floating IP アドレスなしでのインストールの完了

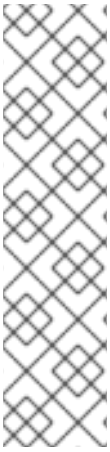
Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

install-config.yaml ファイルで以下のパラメーターを定義しないでください。

- **platform.openstack.ingressFloatingIP**
- **platform.openstack.apiFloatingIP**

外部ネットワークを提供できない場合は、**platform.openstack.externalNetwork** を空白のままにすることもできます。**platform.openstack.externalNetwork** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。外部接続を独自に設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムからインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。



注記

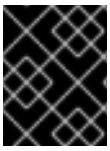
API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

21.5.12. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

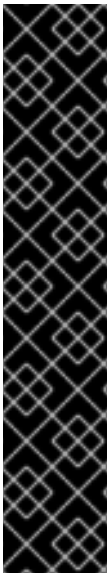
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

21.5.13. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパ

スを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

21.5.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスと理解の詳細については、Web コンソールへのアクセス を参照してください。

21.5.15. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

21.5.16. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

21.5.17. 次のステップ

- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。

- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- Floating IP アドレス上でアプリケーショントラフィックを受け入れるように RHOSP を設定していない場合は、Floating IP アドレスを使用して RHOSP アクセスを設定 します。

21.6. OPENSTACK CLOUD CONTROLLER MANAGER リファレンスガイド

21.6.1. OpenStack Cloud Controller Manager

OpenShift Container Platform 4.12 以降、Red Hat OpenStack Platform (RHOSP) で実行されるクラスターは、従来の OpenStack クラウドプロバイダーから外部の OpenStack Cloud Controller Manager (CCM) に切り替えられました。これは、Kubernetes がツリー内の従来のクラウドプロバイダーから、Cloud Controller Manager を使用して実装される外部クラウドプロバイダーに移行したことに伴う変更です。

従来のクラウドプロバイダーのユーザー定義の設定を保持するために、移行プロセスの一環として、既存の設定が新しい設定にマップされます。**openshift-config** namespaceで **cloud-provider-config** と呼ばれる設定を検索します。

注記

設定マップ名 **cloud-provider-config** は静的に設定されていません。これは、**infrastructure/cluster** CRD の **spec.cloudConfig.name** 値から派生します。

見つかった設定は、**openshift-cloud-controller-manager** namespaceの **cloud-conf** config map に同期されます。

この同期の一環として、OpenStack CCM Operator は新しい config map を変更して、そのプロパティーが外部クラウドプロバイダーと互換性を持つようにします。ファイルは次の方法で変更されます。

- **Global secret-name**、**Global secret-namespace**、および **Global kubeconfig-path** オプションは削除されました。これらは、外部のクラウドプロバイダーには適用されません。
- **Global use-clouds**、**Global clouds-file**、および **Global cloud** オプションが追加されました。
- **[BlockStorage]** セクション全体が削除されます。外部のクラウドプロバイダーは、ストレージ操作を実行しなくなりました。ブロックストレージの設定は、Cinder CSI ドライバーによって管理されます。

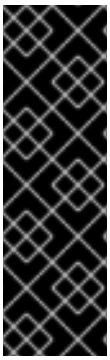
さらに、CCM Operator は多くのデフォルトオプションを適用します。これらのオプションの値は、次のように常にオーバーライドされます。

```
[Global]
use-clouds = true
clouds-file = /etc/openstack/secret/clouds.yaml
cloud = openstack
...

[LoadBalancer]
enabled = true
```

cloud-value 値 **/etc/openstack/secret/clouds.yaml** は、**openshift-cloud-controller-manager** namespace の **openstack-cloud-credentials** 設定にマップされます。他の **clouds.yaml** ファイルと同様に、このファイルで RHOSP クラウドを変更できます。

21.6.2. OpenStack Cloud Controller Manager (CCM) 設定マップ

OpenStack CCM config map は、クラスターが RHOSP クラウドと対話する方法を定義します。デフォルトでは、この設定は **openshift-cloud-controller-manager** namespace にある **cloud-conf** config mapの **cloud.conf** キーの下に保存されます。



重要

cloud-conf config map は、**openshift-config** namespace の **cloud-provider-config** config map から生成されます。

cloud-conf config map で記述されている設定を変更するには、**cloud-provider-config** config map を変更します。

この同期の一環として、CCM Operator はいくつかのオプションをオーバーライドします。詳細は、「RHOSP Cloud Controller Manager」を参照してください。

以下に例を示します。

cloud-conf config map の例

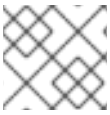
```
apiVersion: v1
data:
  cloud.conf: |
    [Global] 1
    secret-name = openstack-credentials
    secret-namespace = kube-system
    region = regionOne
    [LoadBalancer]
    enabled = True
kind: ConfigMap
metadata:
  creationTimestamp: "2022-12-20T17:01:08Z"
  name: cloud-conf
  namespace: openshift-cloud-controller-manager
  resourceVersion: "2519"
  uid: cbbeedaf-41ed-41c2-9f37-4885732d3677
```

1 config map を変更するのではなく、**clouds.yaml** ファイルを使用してグローバルオプションを設定します。

次のオプションが config map に存在します。特に明記されている場合を除き、RHOSP で実行されるクラスターには必須です。

21.6.2.1. ロードバランサー (オプション)

CCM は、Octavia を使用するデプロイメント用にいくつかのロードバランサーオプションをサポートしています。



注記

Neutron-LBaaS のサポートは廃止されました。

オプション	Description
enabled	**LoadBalancer** タイプのサービス統合を有効にするかどうか。デフォルト値は **true** です。
floating-network-id	オプション:ロードバランサーの仮想 IP アドレス (VIP) のフローティング IP アドレスを作成するために使用される外部ネットワーク。クラウドに複数の外部ネットワークがある場合、このオプションを設定するか、ユーザーがサービスアノテーションで **loadbalancer.openstack.org/floating-network-id** を指定する必要があります。
floating-subnet-id	オプション:ロードバランサー VIP のフローティング IP アドレスを作成するために使用される外部ネットワークサブネット。サービスアノテーション **loadbalancer.openstack.org/floating-subnet-id** でオーバーライドできます。
floating-subnet	オプション:ロードバランサー VIP のフローティング IP アドレスを作成するために使用される外部ネットワークサブネットの名前パターン (~ で始まる場合はグロブまたは正規表現)。サービスアノテーション **loadbalancer.openstack.org/floating-subnet** でオーバーライドできます。複数のサブネットがパターンに一致する場合、使用可能な IP アドレスを持つ最初のサブネットが使用されます。
floating-subnet-tags	オプション:ロードバランサー VIP のフローティング IP アドレスを作成するために使用される外部ネットワークサブネットのタグ。サービスアノテーション **loadbalancer.openstack.org/floating-subnet-tags** でオーバーライドできます。複数のサブネットがこれらのタグに一致する場合、使用可能な IP アドレスを持つ最初のサブネットが使用されます。 RHOSP ネットワークが共有を無効にして設定されている場合 (たとえば、作成時に **--no-share** フラグが使用されている場合)、このオプションはサポートされません。このオプションを使用するには、共有するネットワークを設定します。

オプション	Description
lb-method	ロードバランサープールの作成に使用される負荷分散アルゴリズム。Amphora プロバイダーの場合、値は **ROUND_ROBIN**、**LEAST_CONNECTIONS**、または **SOURCE_IP** です。デフォルト値は **ROUND_ROBIN** です。 OVN プロバイダーでは、**SOURCE_IP_PORT** アルゴリズムのみがサポートされています。 Amphora プロバイダの場合、**LEAST_CONNECTIONS** または **SOURCE_IP** メソッドを使用する場合、**openshift-config** namespece の **cloud-provider-config** config map で **create-monitor** オプションを **true** として設定します。また、ロードバランサータイプサービスの **ETP:Local** で、クライアントからサービスのエンドポイント接続でバランスアルゴリズムの実行ができるよう構成します。
lb-provider	オプション:**amphora** や **octavia** など、ロードバランサーのプロバイダーを指定するために使用されます。Amphora および Octavia プロバイダーのみがサポートされています。
lb-version	オプション:ロードバランサー API のバージョン。**"v2"** のみがサポートされています。
subnet-id	ロードバランサーの仮想 IP が作成される Networking サービスのサブネットの ID。デュアルスタックデプロイメントの場合は、このオプションを未設定のままにしておきます。OpenStack クラウドプロバイダーによって、ロードバランサーに使用するサブネットが自動的に選択されます。
network-id	ロードバランサーの仮想 IP が作成される Networking サービスのネットワークの ID。**subnet-id** が設定されている場合は不要です。このプロパティーが設定されていない場合、ネットワークはクラスターノードが使用するネットワークに基づいて自動的に選択されます。
create-monitor	サービスロードバランサーのヘルスモニターを作成するかどうか。**externalTrafficPolicy: Local** を宣言するサービスには、ヘルスモニターが必要です。デフォルト値は **false** です。 **ovn** プロバイダーでバージョン 17 より前の RHOSP を使用する場合、このオプションはサポートされません。

オプション	Description
monitor-delay	プローブがロードバランサーのメンバーに送信される間隔 (秒単位)。デフォルト値は **5** です。
monitor-max-retries	ロードバランサーメンバーの動作ステータスを **ONLINE** に変更するために必要なチェックの成功回数。有効な範囲は **1** ~ **10** で、デフォルト値は **1** です。
monitor-timeout	モニターがタイムアウトする前にバックエンドへの接続を待機する時間 (秒単位)。デフォルト値は **3** です。
internal-lb	フローティング IP アドレスなしで内部ロードバランサーを作成するかどうか。デフォルト値は **false** です。
LoadBalancerClass "ClassName"	これは、一連のオプションで設定される設定セクションです。 • **floating-network-id** • **floating-subnet-id** • **floating-subnet** • **floating-subnet-tags** • **network-id** • **subnet-id** これらのオプションの動作は、CCM 設定ファイルのロードバランサーセクションにある同じ名前のオプションの動作と同じです。 サービスアノテーション **loadbalancer.openstack.org/class** を指定することで **ClassName** 値を設定できます。
max-shared-lb	ロードバランサーを共有できるサービスの最大数。デフォルト値は **2** です。

21.6.2.2. Operator がオーバーライドするオプション

CCM Operator は、RHOSP の設定で認識できる次のオプションをオーバーライドします。自分で設定しないでください。これらは、情報提供のみを目的としてこのドキュメントに含まれています。

オプション	Description
auth-url	RHOSP ID サービスの URL。たとえば、**http://128.110.154.166/identity** です。

オプション	Description
os-endpoint-type	サービスカタログから使用するエンドポイントのタイプ。
username	Identity サービスのユーザー名。
password	Identity サービスのユーザーパスワード。
domain-id	Identity サービスのユーザードメイン ID。
domain-name	Identity サービスのユーザードメイン名。
tenant-id	Identity サービスのプロジェクト ID。Identity サービスアプリケーションの認証情報を使用している場合は、このオプションを未設定のままにします。 識別子 **tenant** を **project** に変更したバージョン 3 の Identity API では、**tenant-id** の値が API の project コンストラクトに自動的にマップされます。
tenant-name	Identity サービスのプロジェクト名。
tenant-domain-id	Identity サービスプロジェクトのドメイン ID。
tenant-domain-name	Identity サービスプロジェクトのドメイン名。
user-domain-id	Identity サービスのユーザードメイン ID。
user-domain-name	Identity サービスのユーザードメイン名。
use-clouds	**clouds.yaml** ファイルから認証認証情報をフェッチするかどうか。このセクションで設定されたオプションは、**clouds.yaml** ファイルから読み取られた値よりも優先されます。 CCM は、次の場所でファイルを検索します。 1. **clouds-file** オプションの値。 2. 環境変数 **OS_CLIENT_CONFIG_FILE** に格納されているファイルパス。 3. ディレクトリー **pkg/openstack**。 4. ディレクトリー **~/.config/openstack**。 5. ディレクトリー **/etc/openstack**。

オプション	Description
clouds-file	**clouds.yaml** ファイルのファイルパス。**use-clouds** オプションが **true** に設定されている場合に使用されます。
cloud	使用する **clouds.yaml** ファイル内の名前付きクラウド。**use-clouds** オプションが **true** に設定されている場合に使用されます。

21.7. ローカルディスク上の ROOTVOLUME および ETCD を使用した OPENSTACK へのデプロイ

重要

ローカルディスク上の rootVolume と etcd を使用した Red Hat OpenStack Platform (RHOSP) へのデプロイは、テクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

Day 2 オペレーション中に、etcd をルートボリューム (OpenStack Cinder によって提供されるもの) から専用の一時ローカルディスクに移動することで、Red Hat OpenStack Platform (RHOSP) インストールのパフォーマンスの問題を解決および防止できます。

21.7.1. ローカルディスクへの RHOSP のデプロイ

既存の RHOSP クラウドがある場合は、そのクラウドから etcd を専用の一時ローカルディスクに移動できます。

警告

この手順は、ローカルディスク上で etcd をテストすることだけを目的としたものです。実稼働クラスターでは使用しないでください。場合によっては、コントロールプレーンが完全に失われる可能性があります。詳細は、「バックアップおよび復元」の「バックアップおよび復元の操作の概要」を参照してください。

前提条件

- Cinder が動作している OpenStack クラウドがある。

- OpenStack クラウドに、OpenShift コントロールプレーンの 3 つのルートボリュームを収容するために、少なくとも 75 GB の利用可能なストレージがある。
- OpenStack クラウドが、**rbd** ではなくローカルストレージバックエンドを使用する Nova 一時ストレージを使用してデプロイされている。

手順

1. 次のコマンドを実行して、少なくとも 10 GB の一時ディスクを備えたコントロールプレーンの Nova フレーバーを作成します。環境に応じて **--ram**、**--disk**、および <flavor_name> の値を置き換えます。

```
$ openstack flavor create --<ram 16384> --<disk 0> --ephemeral 10 --vcpus 4
<flavor_name>
```

2. コントロールプレーンのルートボリュームを含むクラスターをデプロイします。以下に例を示します。

サンプル YAML ファイル

```
# ...
controlPlane:
  name: master
  platform:
    openstack:
      type: ${CONTROL_PLANE_FLAVOR}
      rootVolume:
        size: 25
        types:
        - ${CINDER_TYPE}
  replicas: 3
# ...
```

3. 次のコマンドを実行して、作成したクラスターをデプロイします。

```
$ openshift-install create cluster --dir <installation_directory> 1
```

1 **<installation_directory>** には、以前に作成したカスタマイズ済みの **./install-config.yaml** ファイルの場所を指定します。

4. 次の手順に進む前に、次のコマンドを実行して、デプロイしたクラスターが正常であることを確認します。

```
$ oc wait clusteroperators --all --for=condition=Progressing=false 1
```

1 クラスター Operator の進行が完了しており、クラスターがデプロイまたは更新中でないことを確認します。

5. 次のコマンドを実行して、**ControlPlaneMachineSet** (CPMS) を編集し、etcd によって使用される追加のブロック一時デバイスを追加します。

```
$ oc patch ControlPlaneMachineSet/cluster -n openshift-machine-api --type json -p ' 1
```

```
[
  {
    "op": "add",
    "path":
"/spec/template/machines_v1beta1_machine_openshift_io/spec/providerSpec/value/additionalB
ockDevices", 2
    "value": [
     {
       "name": "etcd",
       "sizeGiB": 10,
       "storage": {
         "type": "Local" 3
       }
     }
    ]
  }
]
'
```

1 JSON パッチを **ControlPlaneMachineSet** カスタムリソース (CR) に適用します。

2 **AdditionalBlockDevices** を追加するパスを指定します。

3 少なくとも 10 GB のローカルストレージを持つ etcd デバイスをクラスターに追加します。etcd デバイスが Nova フレーバーに収まる限り、10 GB を超える値も指定できます。たとえば、Nova フレーバーの容量が 15 GB の場合、12 GB の etcd デバイスを作成できます。

6. 次の手順を使用して、コントロールプレーンマシンが正常であることを確認します。

   a. 次のコマンドを実行して、コントロールプレーンマシンセットの更新が完了するまで待ちます。

   ```
   $ oc wait --timeout=90m --for=condition=Progressing=false
   controlplanemachineset.machine.openshift.io -n openshift-machine-api cluster
   ```

   b. 次のコマンドを実行して、3 つのコントロールプレーンマシンセットが更新されていることを確認します。

   ```
   $ oc wait --timeout=90m --for=jsonpath='{.status.updatedReplicas}'=3
   controlplanemachineset.machine.openshift.io -n openshift-machine-api cluster
   ```

   c. 次のコマンドを実行して、3 つのコントロールプレーンマシンセットが正常であることを確認します。

   ```
   $ oc wait --timeout=90m --for=jsonpath='{.status.replicas}'=3
   controlplanemachineset.machine.openshift.io -n openshift-machine-api cluster
   ```

   d. 次のコマンドを実行して、クラスター内で **ClusterOperators** が進行中でないことを確認します。

   ```
   $ oc wait clusteroperators --timeout=30m --all --for=condition=Progressing=false
   ```

e. 次のスクリプトを実行して、3 つのコントロールプレーンマシンのそれぞれに、以前に作成した追加のブロックデバイスがあることを確認します。

```
$ cp_machines=$(oc get machines -n openshift-machine-api --
selector='machine.openshift.io/cluster-api-machine-role=master' --no-headers -o custom-
columns=NAME:.metadata.name) 1


if [[ $(echo "${cp_machines}" | wc -l) -ne 3 ]]; then
  exit 1
fi 2


for machine in ${cp_machines}; do
  if ! oc get machine -n openshift-machine-api "${machine}" -o
jsonpath='{.spec.providerSpec.value.additionalBlockDevices}' | grep -q 'etcd'; then
 exit 1
  fi 3
done
```

1 クラスター内で実行されているコントロールプレーンマシンを取得します。

2 **etcd** という名前の **additionalBlockDevices** エントリーを持つマシンに対してイテレートします。

3 **etcd** という名前の **additionalBlockDevice** を持つすべてのコントロールプレーンマシンの名前を出力します。

7. 次の YAML ファイルを使用して、**98-var-lib-etcd.yaml** という名前のファイルを作成します。

警告

この手順は、ローカルディスク上で etcd をテストするためのものです。実稼働クラスターでは使用しないでください。場合によっては、コントロールプレーンが完全に失われる可能性があります。詳細は、「バックアップおよび復元」の「バックアップおよび復元の操作の概要」を参照してください。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 98-var-lib-etcd
spec:
  config:
    ignition:
      version: 3.4.0
    systemd:
```

```
    units:
    - contents: |
        [Unit]
        Description=Mount local-etcd to /var/lib/etcd

        [Mount]
        What=/dev/disk/by-label/local-etcd (1)
        Where=/var/lib/etcd
        Type=xfs
        Options=defaults,prjquota

        [Install]
        WantedBy=local-fs.target
      enabled: true
      name: var-lib-etcd.mount
    - contents: |
        [Unit]
        Description=Create local-etcd filesystem
        DefaultDependencies=no
        After=local-fs-pre.target
        ConditionPathIsSymbolicLink=!/dev/disk/by-label/local-etcd (2)

        [Service]
        Type=oneshot
        RemainAfterExit=yes
        ExecStart=/bin/bash -c "[ -L /dev/disk/by-label/ephemeral0 ] || ( >&2 echo Ephemeral
disk does not exist; /usr/bin/false )"
        ExecStart=/usr/sbin/mkfs.xfs -f -L local-etcd /dev/disk/by-label/ephemeral0 (3)

        [Install]
        RequiredBy=dev-disk-by\x2dlabel-local\x2detcd.device
      enabled: true
      name: create-local-etcd.service
    - contents: |
        [Unit]
        Description=Migrate existing data to local etcd
        After=var-lib-etcd.mount
        Before=crio.service (4)

        Requisite=var-lib-etcd.mount
        ConditionPathExists=!/var/lib/etcd/member
        ConditionPathIsDirectory=/sysroot/ostree/deploy/rhcos/var/lib/etcd/member (5)

        [Service]
        Type=oneshot
        RemainAfterExit=yes

        ExecStart=/bin/bash -c "if [ -d /var/lib/etcd/member.migrate ]; then rm -rf
/var/lib/etcd/member.migrate; fi" (6)

        ExecStart=/usr/bin/cp -aZ /sysroot/ostree/deploy/rhcos/var/lib/etcd/member/
/var/lib/etcd/member.migrate
        ExecStart=/usr/bin/mv /var/lib/etcd/member.migrate /var/lib/etcd/member (7)

        [Install]
```

```
      RequiredBy=var-lib-etcd.mount
    enabled: true
    name: migrate-to-local-etcd.service
  - contents: |
      [Unit]
      Description=Relabel /var/lib/etcd

      After=migrate-to-local-etcd.service
      Before=crio.service

      [Service]
      Type=oneshot
      RemainAfterExit=yes

      ExecCondition=/bin/bash -c "[ -n \"$(restorecon -nv /var/lib/etcd)\" ]" 8

      ExecStart=/usr/sbin/restorecon -R /var/lib/etcd

      [Install]
      RequiredBy=var-lib-etcd.mount
    enabled: true
    name: relabel-var-lib-etcd.service
```

1 etcd データベースは、ラベルではなくデバイスによってマウントする必要があります。この設定で使用されるデバイスの依存関係を **systemd** に生成させ、ファイルシステムの作成をトリガーするためです。

2 ファイルシステム **dev/disk/by-label/local-etcd** がすでに存在する場合は実行しないでください。

3 **/dev/disk/by-label/ephemeral0** が存在しない場合は、警告メッセージが表示されて失敗します。

4 既存のデータをローカル etcd データベースに移行します。この設定では、**/var/lib/etcd** がマウントされた後、CRI-O が起動する前に、つまり etcd がまだ実行されていないときにデータを移行します。

5 etcd をマウントすること、etcd にメンバーディレクトリーを含めないこと、および ostree にメンバーディレクトリーを含めることを必須とします。

6 以前の移行状態をクリーンアップします。

7 コピーと移動を別々のステップで実行し、完全なメンバーディレクトリーの作成をアトミック操作として実行します。

8 完全な再帰的なラベルの再設定を実行する前に、マウントポイントディレクトリーの簡単なチェックを実行します。ファイルパス **/var/lib/etcd** 内の restorecon がディレクトリーの名前を変更できない場合、再帰的な名前変更が実行されません。

8. 次のコマンドを実行して、新しい **MachineConfig** オブジェクトを作成します。

```
$ oc create -f 98-var-lib-etcd.yaml
```

注記

etcd データベースを各コントロールプレーンマシンのローカルディスクに移動するには時間がかかります。

9. 次のコマンドを実行して、etcd データベースが各コントロールプレーンのローカルディスクに転送されたことを確認します。

   a. 次のコマンドを実行して、クラスターがまだ更新中であることを確認します。

   ```
   $ oc wait --timeout=45m --for=condition=Updating=false machineconfigpool/master
   ```

   b. 次のコマンドを実行して、クラスターの準備ができていることを確認します。

   ```
   $ oc wait node --selector='node-role.kubernetes.io/master' --for condition=Ready --timeout=30s
   ```

   c. 次のコマンドを実行して、クラスター Operators がクラスター内で実行されていることを確認します。

   ```
   $ oc wait clusteroperators --timeout=30m --all --for=condition=Progressing=false
   ```

21.7.2. 関連情報

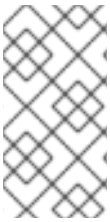
- etcd についての推奨されるプラクティス
- バックアップおよび復元オプションの概要

21.8. OPENSTACK でのクラスターのアンインストール

Red Hat OpenStack Platform (RHOSP) にデプロイしたクラスターを削除できます。

21.8.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。



注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

21.9. 独自のインフラストラクチャーからの RHOSP のクラスターのアンインストール

ユーザーによってプロビジョニングされたインフラストラクチャーの Red Hat OpenStack Platform (RHOSP) にデプロイしたクラスターを削除することができます。

21.9.1. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでの削除プロセスを簡素化する Ansible Playbook には、複数の Python モジュールが必要です。プロセスを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。

注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされている。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk
```

3. **python** コマンドが **python3** を参照していることを確認します。

```
$ sudo alternatives --set python /usr/bin/python3
```

21.9.2. 独自のインフラストラクチャーを使用する RHOSP からのクラスターの削除

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) の OpenShift Container Platform クラスターを削除できます。削除プロセスを迅速に完了するには、複数の Ansible Playbook を実行します。

前提条件

- Python 3 がマシンにインストールされている。
- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- クラスターのインストールに使用した Playbook がある。
- 対応するインストール Playbook に加えた変更を反映するように **down-** の接頭辞が付けられた Playbook を変更している。たとえば、**bootstrap.yaml** ファイルへの変更は **down-bootstrap.yaml** ファイルに反映されます。
- すべての Playbook は共通ディレクトリーにある。

手順

1. コマンドラインで、ダウンロードした Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml  \
 down-bootstrap.yaml      \
 down-control-plane.yaml  \
 down-compute-nodes.yaml  \
 down-load-balancers.yaml \
 down-network.yaml        \
 down-security-groups.yaml
```

2. OpenShift Container Platform インストールに対して加えた DNS レコードの変更を削除します。

OpenShift Container Platform はお使いのインフラストラクチャーから削除されます。

21.10. OPENSTACK のインストール設定パラメーター

OpenShift Container Platform クラスターを Red Hat OpenStack Platform (RHOSP) にデプロイする前に、パラメーターを指定してクラスターとそれをホストするプラットフォームをカスタマイズします。**install-config.yaml** ファイルを作成するときは、コマンドラインを使用して必要なパラメーターの値を指定します。その後、**install-config.yaml** ファイルを変更して、クラスターをさらにカスタマイズできます。

21.10.1. OpenStack で使用可能なインストール設定パラメーター

以下の表に、インストールプロセスの一部として設定できる必須、オプション、および OpenStack 固有のインストール設定パラメーターを示します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

21.10.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表21.6 必須パラメーター

パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

21.10.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表21.7 ネットワークパラメーター

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

21.10.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表21.8 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String

パラメーター	説明	値
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
compute: hyperthreading:	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
controlPlane: hyperthreading:	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**

パラメーター	説明	値
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。	**false** または **true**

パラメーター	説明	値
	重要  クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。	
imageContentSources:	 release-image コンテナーのインストールおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	 **imageContentSources** を使用する場合は必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	 **注記** 同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。 Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	文字列の配列。

パラメーター	説明	値
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。  **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

21.10.1.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表21.9 追加の RHOSP パラメーター

パラメーター	説明	値
compute: platform: openstack: rootVolume: size:	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute: platform: openstack: rootVolume: types:	コンピュートマシンの場合は、root のボリュームタイプです。	文字列のリスト (例: {**performance-host1**, **performance-host2**, **performance-host3**})。[1]
compute: platform: openstack: rootVolume: type:	コンピュートマシンの場合、root のボリュームタイプです。このプロパティーは非推奨となり、**compute.platform.openstack.rootVolume.types** に置き換えられます。	文字列 (例: **performance**)。[2]
compute: platform: openstack: rootVolume: zones:	コンピュートマシンの場合、ルートボリュームをインストールする Cinder 可用性ゾーン。このパラメーターに値を設定しない場合、インストールプログラムはデフォルトのアベイラビリティーゾーンを選択します。このパラメーターは、**compute.platform.openstack.zones** が定義されている場合には必須です。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
controlPlane: platform: openstack: rootVolume: size:	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。

パラメーター	説明	値
controlPlane: platform: openstack: rootVolume: types:	コントロールプレーンマシンの場合は、root ボリュームのタイプです。	文字列のリスト (例: {**performance-host1**, **performance-host2**, **performance-host3**})。[1]
controlPlane: platform: openstack: rootVolume: type:	コントロールプレーンマシンの場合、root ボリュームのタイプです。このプロパティーは非推奨となり、**compute.platform.openstack.rootVolume.types** に置き換えられます。	文字列 (例: **performance**)。[2]
controlPlane: platform: openstack: rootVolume: zones:	コントロールプレーンマシンの root ボリュームをインストールする Cinder アベイラビリティーゾーン。この値を設定しない場合、インストールプログラムはデフォルトのアベイラビリティーゾーンを選択します。**controlPlane.platform.openstack.zones** が定義されている場合、このパラメーターは必須です。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
platform: openstack: cloud:	**clouds.yaml** ファイルのクラウドリストにある使用する RHOCP クラウドの名前。 可能であれば、**clouds.yaml** ファイルのクラウド設定では、ユーザー名とパスワードの組み合わせではなく、アプリケーションの認証情報を使用します。アプリケーション認証情報を使用すると、ユーザー名とパスワードのローテーションに伴うシークレットの伝達による中断を回避できます。	文字列 (例: **MyCloud**)。

パラメーター	説明	値
platform: openstack: externalNetwork:	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform: openstack: computeFlavor:	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。 このプロパティーは非推奨にされています。すべてのマシンプールのデフォルトとしてフレーバーを使用するには、これを **platform.openstack.defaultMachinePlatform** プロパティーで **type** キーの値として追加します。それぞれのマシンプールのフレーバー値を個別に設定することもできます。	文字列 (例: **m1.xlarge**)。

1. マシンプールが **zones** を定義している場合、タイプの数は 1 つの項目であるか、**zones** 内の項目の数と一致できます。たとえば、**zones** に項目が 3 つある場合は、タイプの数を 2 にすることはできません。

2. このプロパティーへの既存の参照がある場合、インストーラーは、**controlPlane.platform.openstack.rootVolume.types** フィールドに対応する値を設定します。

21.10.1.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表21.10 オプションの RHOSP パラメーター

パラメーター	説明	値
compute: platform: openstack: additionalNetworkIDs:	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID のリスト。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

パラメーター	説明	値
compute: platform: openstack: additionalSecurityGroupIDs:	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID のリスト。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
compute: platform: openstack: zones:	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストールプログラムは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。	文字列のリスト例: **["zone-1", "zone-2"]**。
compute: platform: openstack: serverGroupPolicy:	プール内のコンピュートマシンを含むグループに適用するサーバーグループポリシー。作成後にサーバーグループのポリシーまたは所属を変更することはできません。サポートされているオプションには、**anti-affinity**、**soft-affinity**、および **soft-anti-affinity** が含まれます。デフォルト値は **soft-anti-affinity** です。 **affinity** ポリシーは移行を防止するため、RHOSP のアップグレードに影響します。**affinity** ポリシーはサポートされていません。 厳密な **anti-affinity** ポリシーを使用する場合は、インスタンスの移行中に追加の RHOSP ホストが必要です。	マシンプールに適用するサーバーグループポリシー。たとえば、**soft-affinity**。

パラメーター	説明	値
controlPlane: platform: openstack: additionalNetworkIDs:	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。 コントロールプレーンマシンに接続されている追加のネットワークも、ブートストラップノードに接続されています。	文字列としての 1 つ以上の UUID のリスト。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
controlPlane: platform: openstack: additionalSecurityGroupIDs:	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID のリスト。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane: platform: openstack: zones:	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストールプログラムは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。	文字列のリスト例: **["zone-1", "zone-2"]**。

パラメーター	説明	値
controlPlane: platform: openstack: serverGroupPolicy:	プール内のコントロールプレーンマシンを含むグループに適用するサーバーグループポリシー。作成後にサーバーグループのポリシーまたは所属を変更することはできません。サポートされているオプションには、**anti-affinity**、**soft-affinity**、および **soft-anti-affinity** が含まれます。デフォルト値は **soft-anti-affinity** です。 **affinity** ポリシーは移行を防止するため、RHOSP のアップグレードに影響します。**affinity** ポリシーはサポートされていません。 厳密な **anti-affinity** ポリシーを使用する場合は、インスタンスの移行中に追加の RHOSP ホストが必要です。	マシンプールに適用するサーバーグループポリシー。たとえば、**soft-affinity**。
platform: openstack: clusterOSImage:	インストールプログラムが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

パラメーター	説明	値
platform: openstack: clusterOSImageProperties:	Glance のインストーラーでアップロードされた ClusterOSImage に追加するプロパティー。このプロパティーは、**platform.openstack.clusterOSImage** が既存の Glance イメージに設定されている場合は無視されます。 このプロパティーを使用し、ノードあたり 26 PV の RHOSP のデフォルト永続ボリューム (PV) の制限を超過することができます。制限を超えるには、**hw_scsi_model** プロパティーの値を **virtio-scsi** に設定し、**hw_disk_bus** の値を **scsi** に設定します。 このプロパティーを使用し、**hw_qemu_guest_agent** プロパティーを **yes** の値で追加して QEMU ゲストエージェントを有効にすることもできます。	キーと値の文字列のペアのリスト。例: **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**
platform: openstack: defaultMachinePlatform:	デフォルトのマシンプールプラットフォームの設定。	<pre>{ "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>
platform: openstack: ingressFloatingIP:	Ingress ポートに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。

パラメーター	説明	値
platform: openstack: apiFloatingIP:	API ロードバランサーに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。
platform: openstack: externalDNS:	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスのリスト。例: **["8.8.8.8", "192.168.1.12"]**
platform: openstack: loadbalancer:	デフォルトの内部ロードバランサーを使用するかどうか。値が **UserManaged** に設定されている場合、このデフォルトのロードバランサーは無効になり、外部のユーザー管理のロードバランサーを使用するクラスターをデプロイできるようになります。パラメーターが設定されていない場合、または値が **OpenShiftManagedDefault** の場合、クラスターはデフォルトのロードバランサーを使用します。	**UserManaged** または **OpenShiftManagedDefault**。
platform: openstack: machinesSubnet:	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

パラメーター	説明	値

第22章 OCI へのインストール

22.1. ASSISTED INSTALLER を使用して ORACLE CLOUD INFRASTRUCTURE (OCI) にクラスターをインストールする

OpenShift Container Platform 4.16 以降のバージョンでは、独自に提供するインフラストラクチャーを使用して、Assisted Installer を使用して Oracle® Cloud Infrastructure (OCI) にクラスターをインストールできます。

22.1.1. Assisted Installer と OCI の概要

専用、ハイブリッド、パブリックおよびマルチクラウド環境をサポートする Oracle® Cloud Infrastructure (OCI) インフラストラクチャー上でクラスターワークロードを実行できます。Red Hat と Oracle はどちらも、OCI 上の OpenShift Container Platform クラスターでの OCI の実行をテスト、検証、サポートしています。

Assisted Installer は OCI プラットフォームをサポートしています。Assisted Installer では、OCI へのクラスターのインストールタスクを自動化するために、直感的な対話型ワークフローを利用できます。

OCI は、規制コンプライアンス、パフォーマンス、費用対効果のニーズを満たすサービスを提供します。OCI Resource Manager 設定にアクセスして、OCI リソースをプロビジョニングおよび設定できます。

重要

OCI リソースをプロビジョニングする手順は、例としてのみ提供されています。他の方法で必要なリソースを作成することも選択できます。スクリプトは単なる例です。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。OCI Resource Manager 設定にアクセスして以下の手順を完了することも、設定をモデルとして使用して独自のカスタムスクリプトを作成することもできます。

Assisted Installer を使用して OpenShift Container Platform クラスターを OCI にインストールする方法を理解するには、**Assisted Installer を使用して Oracle Cloud Infrastructure (OCI) にクラスターをインストールする** ドキュメントの手順に従ってください。このドキュメントでは、OCI Cloud Controller Manager (CCM) および Oracle の Container Storage Interface (CSI) オブジェクトを使用して OpenShift Container Platform クラスターを OCI API にリンクする方法を示します。

重要

OCI 上で動作するクラスターワークロードに最適なパフォーマンス条件を確保するために、ブロックボリュームのボリュームパフォーマンスユニット (VPU) がワークロードに適したサイズに設定されていることを確認してください。次のリストを参考に、特定のパフォーマンスニーズに応じて必要な VPU を選択してください。

- テストまたは概念実証環境: 100 GB、20 - 30 VPU。
- 基本的な環境: 500 GB、60 VPU
- 高負荷の実稼働環境: 500 GB 以上、100 以上の VPU

更新やスケーリングアクティビティーに十分な容量を提供できるように、VPU を余分に確保しておくことを検討してください。VPU の詳細は、Oracle ドキュメントの「Volume Performance Units」を参照してください。

OpenShift Container Platform の Assisted Installer に慣れていない場合は、「OpenShift Container Platform の Assisted Installer」を参照してください。

関連情報

- OpenShift Container Platform のアシステッドインストーラー
- OpenShift Container Platform のインターネットアクセス
- Volume Performance Units (Oracle ドキュメント)
- OCI ノード上の OpenShift Container Platform のインスタンスサイズ設定の推奨事項 (Oracle) ドキュメント

22.1.2. OCI リソースとサービスの作成

Oracle® Cloud Infrastructure (OCI) のリソースとサービスを作成すると、組織の要件を満たすガバナンス標準に準拠したインフラストラクチャーを確立できます。

前提条件

- クラスターをホストするための OCI アカウントを設定した。Prerequisites (Oracle ドキュメント) を参照してください。

手順

1. 管理者権限を使用して Oracle Cloud Infrastructure (OCI) アカウントにログインします。
2. Oracle リソースからアーカイブファイルをダウンロードします。アーカイブファイルには、クラスターリソースとカスタムマニフェストを作成するためのファイルが含まれています。アーカイブファイルにはスクリプトも含まれており、スクリプトを実行すると、DNS レコードやインスタンスなどの OCI リソースが作成されます。詳細は、Configuration Files (Oracle ドキュメント) を参照してください。

22.1.3. Assisted Installer を使用して OCI 互換の検出 ISO イメージを生成する

検出 ISO イメージを生成し、そのイメージを Oracle® Cloud Infrastructure (OCI) にアップロードすると、エージェントによりハードウェアとネットワークの検証チェックを実行してから、OpenShift Container Platform クラスターを OCI にインストールできます。

OCI Web コンソールから、次のリソースを作成する必要があります。

- OCI リソースをより適切に整理し、アクセスを制限し、使用制限を設定するためのコンパートメント。
- 検出 ISO イメージを安全かつ確実に保存するためのオブジェクトストレージバケット。後の段階でインスタンスを起動するためにイメージにアクセスして、クラスターを作成できます。

前提条件

- OCI 上に子コンパートメントとオブジェクトストレージバケットを作成した。Oracle ドキュメントの Provisioning Cloud Infrastructure (OCI Console) を参照してください。
- OpenShift Container Platform のインストールおよび更新プロセスの詳細を確認している。
- ファイアウォールを使用しており、Telemetry サービスを使用する予定の場合は、OpenShift Container Platform が必要なサイトにアクセスできるようにファイアウォールを設定しておく必要があります。
- 仮想マシン (VM) を作成する前に、クラウドインスタンスタイプ (Red Hat Ecosystem Catalog ポータル) を参照して、サポートされている OCI 仮想マシンシェイプを特定してください。

手順

1. Hybrid Cloud Console の **Install OpenShift with the Assisted Installer** ページから、Assisted Installer の必要な手順をすべて完了して検出 ISO イメージを生成します。
   a. **Cluster Details** ステップで、次のフィールドに値を入力します。

フィールド	必要な操作
Cluster name	クラスターの名前 (**ocidemo** など) を指定します。
Base domain	クラスターのベースドメイン (**splat-oci.devcluster.openshift.com** など) を指定します。以前に OCI でコンパートメントを作成している場合は、**DNS management → Zones → List scope** に移動し、親コンパートメントを選択することで、この情報を取得できます。ベースドメインが **Public zones** タブに表示されます。
OpenShift version	**OpenShift 4.15** 以降のバージョンを指定します。
CPU アーキテクチャー	**x86_64** または **Arm64** を指定します。

フィールド	必要な操作
Integrate with external partner platforms	**Oracle Cloud Infrastructure** を指定します。 この値を指定すると、デフォルトで **Include custom manifests** チェックボックスが選択されます。

b. **Operators** ページで、**Next** をクリックします。

c. **Host Discovery** ページで、**Add hosts** をクリックします。

d. **SSH public key** フィールドに、ローカルシステムから SSH キーを追加します。

ヒント

ssh-keygen ツールを使用して、SSH 認証キーペアを作成できます。

e. **Generate Discovery ISO** をクリックして、検出 ISO イメージファイルを生成します。

f. ローカルシステムにファイルをダウンロードします。

2. 検出 ISO イメージを OCI バケットにアップロードします。Uploading an Object Storage Object to a Bucket (Oracle ドキュメント) を参照してください。

a. アップロードされた検出 ISO イメージに対して、事前認証されたリクエストを作成する必要があります。後で OCI スタックを作成するときに URL を指定する必要があるため、事前認証されたリクエストの URL を必ずメモしておいてください。

関連情報

- インストールおよび更新
- ファイアウォールの設定

22.1.4. クラスター用の OCI インフラストラクチャーのプロビジョニング

Assisted Installer を使用して OpenShift Container Platform クラスターの詳細を作成することで、スタックでこれらの詳細を指定できます。スタックは、OpenShift Container Platform クラスターを OCI にインストールするのに必要なすべての OCI インフラストラクチャーリソース (カスタムイメージなど) のプロビジョニングを自動化できる OCI の機能です。

Oracle® Cloud Infrastructure (OCI) Compute サービスは、OCI 上に仮想マシン (VM) インスタンスを作成します。このインスタンスは、仮想クラウドネットワーク (VNC) サブネット内の仮想ネットワークインターフェイスコントローラー (vNIC) に自動的に接続できます。カスタムマニフェストテンプレートファイルで OpenShift Container Platform クラスターの IP アドレスを指定すると、OCI インスタンスが VNC 経由でクラスターと通信できるようになります。

前提条件

- 検出 ISO イメージを OCI バケットにアップロードしている。詳細は、「Assisted Installer を使用して OCI 互換の検出 ISO イメージを生成する」を参照してください。

手順

1. OpenShift Container Platform クラスターの OCI インフラストラクチャーをプロビジョニングする手順を完了します。Creating OpenShift Container Platform Infrastructure Using Resource Manager (Oracle ドキュメント) を参照してください。

2. スタックを作成し、Editing the OpenShift Custom Manifests (Oracle ドキュメント) の手順に従ってカスタムマニフェストファイルを編集します。

22.1.5. Assisted Installer の残りの手順を完了する

Oracle® Cloud Infrastructure (OCI) リソースをプロビジョニングし、OpenShift Container Platform カスタムマニフェスト設定ファイルを OCI にアップロードした後、インスタンス OCI を作成する前に、Assisted Installer で残りのクラスターのインストール手順を完了する必要があります。

前提条件

- カスタムマニフェスト設定ファイルと OCI リソースマネージャー設定リソースを含むリソーススタックを OCI 上に作成した。「クラスター用の OCI インフラストラクチャーのプロビジョニング」を参照してください。

手順

1. Red Hat Hybrid Cloud Console Web コンソールから、**Host discovery** ページに移動します。

2. **Role** 列で、対象のホスト名ごとに **Control plane node** または **Worker** を選択します。

重要

次のステップに進む前に、各ノードが **Ready** ステータスに達するまで待ちます。

3. **Storage** ステップと **Networking** ステップのデフォルト設定を受け入れ、**Next** をクリックします。

4. **Custom manifests** ページの **Folder** フィールドで、**manifest** を選択します。これは、カスタムマニフェストファイルを保存する Assisted Installer 用のフォルダーです。

   a. **File name** フィールドに、**oci-ccm.yml** などの値を入力します。

   b. **Content** セクションで **Browse** をクリックし、**custom_manifest/manifests/oci-ccm.yml** にあるドライブから CCM マニフェストを選択します。

5. 次の **Custom manifest** セクションを展開し、以下のマニフェストに対して同じ手順を繰り返します。

   - CSI ドライバーマニフェスト: **custom_manifest/manifests/oci-csi.yml**
   - CCM マシン設定: **custom_manifest/openshift/machineconfig-ccm.yml**
   - CSI ドライバーマシン設定: **custom_manifest/openshift/machineconfig-csi.yml**

6. **Review and create** ページで、**Install cluster** をクリックして、OCI 上に OpenShift Container Platform クラスターを作成します。

クラスターのインストールと初期化操作が完了すると、Assisted Installer によってクラスターのインストール操作が完了したことが示されます。詳細は、**OpenShift Container Platform の Assisted Installer** ドキュメントの「インストールの完了」セクションを参照してください。

関連情報

- OpenShift Container Platform のアシステッドインストーラー

22.1.6. OCI へのクラスターのインストールが成功したことを確認する

クラスターがインストールされ、Oracle® Cloud Infrastructure (OCI) 上で効果的に実行されていることを確認します。

手順

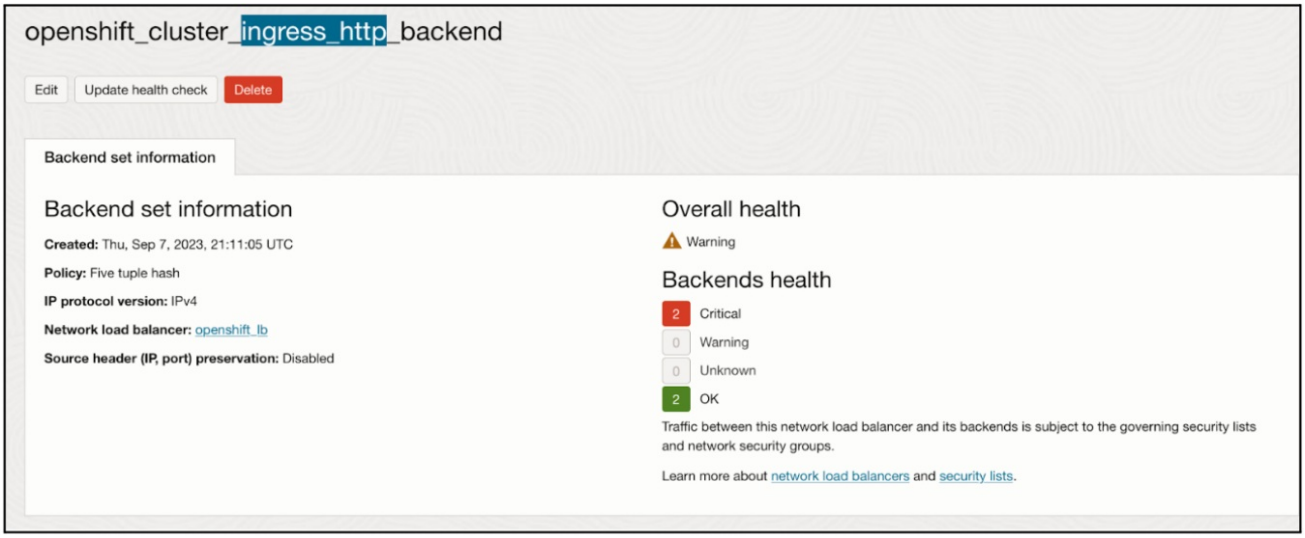
1. Hybrid Cloud Console から **Clusters** > **Assisted Clusters** に移動し、クラスターの名前を選択します。
2. インストールの進行状況バーが 100% になっていて、“Installation completed successfully” というメッセージが表示されていることを確認します。
3. OpenShift Container Platform Web コンソールにアクセスするために、表示される Web コンソール URL をクリックします。
4. **Nodes** メニューページに移動します。
5. **Nodes** テーブルからノードを見つけます。
6. **Overview** タブで、ノードが **Ready** ステータスになっていることを確認します。
7. YAML タブを選択します。
8. **labels** パラメーターを確認し、表示されるラベルが、使用する設定に該当するものであることを確認します。たとえば、**topology.kubernetes.io/region=us-sanjose-1** というラベルからは、ノードがデプロイされた OCI リージョンがわかります。

22.1.7. OCI でのクラスターのインストールのトラブルシューティング

Assisted Installer を使用して OpenShift Container Platform クラスターを Oracle® Cloud Infrastructure (OCI) にインストールする際に問題が発生した場合は、次のセクションを読んで一般的な問題のトラブルシューティングを行ってください。

OCI の Ingress ロードバランサーが健全なステータスでない
この問題は、**Warning** として分類されています。Resource Manager を使用してスタックを作成すると、デフォルトで 3 つのコンピュートノードのプールが作成され、Ingress ロードバランサーのバックエンドリスナーとして自動的に追加されるためです。デフォルトでは、OpenShift Container Platform は 2 つのルーター Pod をデプロイします。これらの Pod は、OpenShift Container Platform マニフェストファイルのデフォルト値に基づいています。3 つのコンピュートノードで実行できる使用可能なルーター Pod の数 (2 つ) との不一致により、このような **Warning** が発生することが予想されます。

図22.1 OCI の Backend set information タブに表示されるWarning メッセージの例:



Ingress ロードバランサーの設定を変更する必要はありません。代わりに、OpenShift Container Platform 上のクラスター内で動作する特定のコンピュートノードに、Ingress ロードバランサーを参照させることができます。これを行うには、OpenShift Container Platform でアノテーションなどの配置メカニズムを使用して、最初に Ingress ロードバランサーにバックエンドリスナーとして設定したコンピュートノード上でのみルーター Pod を実行します。

OCI スタック作成操作が失敗し、Error: 400-InvalidParameter というメッセージが表示される
OCI でスタックを作成しようとすると、ジョブの **Logs** セクションにエラーメッセージが出力されることがあります。以下に例を示します。

> Error: 400-InvalidParameter, DNS Label oci-demo does not follow Oracle requirements
> Suggestion: Please update the parameter(s) in the Terraform config as per error message DNS Label oci-demo does not follow Oracle requirements
> Documentation: https://registry.terraform.io/providers/oracle/oci/latest/docs/resources/core_vcn

Hybrid Cloud Console の **Install OpenShift with the Assisted Installer** ページに移動し、**Cluster Details** ステップの **Cluster name** フィールドを確認します。ハイフン (**-**) などの特殊文字は、OCI 命名規則に対応性してないため、名前から削除してください。たとえば、**oci-demo** は **ocidemo** に変更します。

関連情報

- Troubleshooting OpenShift Container Platform on OCI (Oracle ドキュメント)
- Assisted Installer を使用したオンプレミスクラスターのインストール

22.2. AGENT-BASED INSTALLER を使用して ORACLE CLOUD INFRASTRUCTURE (OCI) にクラスターをインストールする

OpenShift Container Platform 4.16 では、Agent-based Installer を使用して Oracle® Cloud Infrastructure (OCI) にクラスターをインストールすると、専用、ハイブリッド、パブリックおよびマルチクラウド環境をサポートするインフラストラクチャー上でクラスターのワークロードを実行できます。

22.2.1. Agent-based Installer と OCI の概要

Agent-based Installer を使用して、OpenShift Container Platform クラスターを Oracle® Cloud

Infrastructure (OCI) にインストールできます。Red Hat と Oracle はどちらも、OCI 上の OpenShift Container Platform クラスターで OCI および Oracle® Cloud VMware Solution (OCVS) ワークロードの実行をテスト、検証、サポートしています。

Agent-based Installer は、Assisted Installation サービスを使いやすくするだけでなく、接続環境または非接続環境のいずれかにクラスターをインストールする機能を備えています。

OCI は、規制コンプライアンス、パフォーマンス、費用対効果のニーズを満たすサービスを提供します。OCI は、64 ビット **x86** インスタンスと 64 ビット **ARM** インスタンスをサポートします。さらに、OCI は、アプリケーションのアーキテクチャーの変更を最小限に抑えながら VMware ワークロードを OCI に移動できる OCVS サービスを提供します。

注記

ブートディスクには、Nonvolatile Memory Express (NVMe) ドライブまたはソリッドステートドライブ (SSD) を選択することを検討してください。これらのドライブは、低遅延機能と高スループット機能をブートディスクに提供するためです。

OCI で OpenShift Container Platform クラスターを実行すると、次の機能を利用できます。

- コンピュートフレキシブルシェイプ。仮想マシンの Oracle® CPU (OCPU) とメモリーリソースの数をカスタマイズできます。この機能を利用すると、リソースのバランスが取れた環境でクラスターのワークロードが操作を実行できます。Red Hat Ecosystem Catalog ポータルの Oracle ページにアクセスすると、RHEL 認定を受けたすべての OCI シェイプを参照できます。
- ブロックボリュームストレージ。ストレージボリュームのスケーリングと自動チューニングを設定できるため、ブロックボリュームサービスによりパフォーマンスレベルを自動的に調整してパフォーマンスを最適化できます。
- OCVS。VMware® vSphere ソフトウェア定義データセンタ (SDDC) 上で動作するパブリッククラウド環境にクラスターをデプロイできます。VMware vSphere 環境に対する完全な管理権限を維持しながら、OCI サービスを使用して、柔軟かつスケーラブルでセキュアなインフラストラクチャー上でアプリケーションを改善できます。

重要

OCI および OCVS サービス上で動作するクラスターワークロードに最適なパフォーマンス条件を確保するために、ブロックボリュームのボリュームパフォーマンスユニット (VPU) がワークロードに適したサイズに設定されていることを確認してください。次のリストを参考に、特定のパフォーマンスニーズに応じて必要な VPU を選択してください。

- テストまたは概念実証環境: 100 GB、20 - 30 VPU。
- 基本的な環境: 500 GB、60 VPU
- 高負荷の実稼働環境: 500 GB 以上、100 以上の VPU

更新やスケーリングアクティビティーに十分な容量を提供できるように、VPU を余分に確保しておくことを検討してください。VPU の詳細は、Oracle ドキュメントの「Volume Performance Units」を参照してください。

関連情報

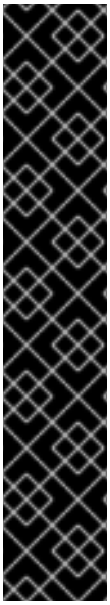
- インストールプロセス

- OpenShift Container Platform のインターネットアクセス
- Agent-based Installer について
- Overview of the Compute Service (Oracle ドキュメント)
- Volume Performance Units (Oracle ドキュメント)
- Instance Sizing Recommendations for OpenShift Container Platform on OCI Nodes (Oracle ドキュメント)

22.2.2. OCI インフラストラクチャーのリソースとサービスの作成

仮想マシン (VM) シェイプ上に OCI 環境を作成する必要があります。この環境を作成すると、OpenShift Container Platform をインストールし、幅広いクラウドオプションと強力なセキュリティーポリシーをサポートするインフラストラクチャーにクラスターをデプロイできます。OCI コンポーネントに関する事前の知識があれば、OCI リソースの概念を理解し、組織のニーズに合わせてリソースを設定する方法を理解するのに役立ちます。

OCI に OpenShift Container Platform クラスターをインストールするためのエージェントベースのインストーラー方式では、OCI リソースとサービスを手動で作成する必要があります。



重要

OpenShift Container Platform との互換性を確保するには、各 DNS レコードのレコードタイプとして **A** を設定し、次のようにレコードに名前を付ける必要があります。

- **api.<cluster_name>.<base_domain>**。これは API ロードバランサーの **apiVIP** パラメーターをターゲットとしています。
- **api-int.<cluster_name>.<base_domain>**。これは API ロードバランサーの **apiVIP** パラメーターをターゲットとしています。
- ***.apps.<cluster_name>.<base_domain>**。これは Ingress ロードバランサーの **ingressVIP** パラメーターをターゲットとしています。

api.* および **api-int.*** DNS レコードは、コントロールプレーンマシンに関連しています。そのため、インストールした OpenShift Container Platform クラスター内のすべてのノードがこれらの DNS レコードにアクセスできることを確認する必要があります。

前提条件

- OpenShift Container Platform クラスターをホストするために OCI アカウントを設定しました。Prerequisites (Oracle ドキュメント) を参照してください。

手順

- 必要な OCI リソースとサービスを作成します。OCI Resources Needed for Using the Agent-based Installer (Oracle ドキュメント) を参照してください。

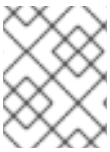
関連情報

- Learn About Oracle Cloud Basics (Oracle ドキュメント)

22.2.3. OCI にクラスターをインストールするための設定ファイルの作成

Agent-based Installer を使用して起動可能な ISO イメージを生成できるように、**install-config.yaml** 設定ファイルと **agent-config.yaml** 設定ファイルを作成する必要があります。エージェントベースのインストールは、Assisted Discovery Agent と Assisted Service を含む起動可能な ISO で構成されます。これらのコンポーネントは、クラスターのインストールを実行するために両方とも必要ですが、後者のコンポーネントはいずれか 1 つのホストでのみ実行されます。

後の段階では、Oracle ドキュメントの手順に従って、生成されたエージェント ISO イメージを Oracle のデフォルトの Object Storage バケットにアップロードする必要があります。これは、OpenShift Container Platform クラスターを Oracle® Cloud Infrastructure (OCI) に統合するための最初の手順です。



注記

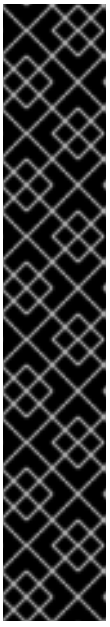
Agent-based Installer を使用して、ゼロタッチプロビジョニング (ZTP) カスタムリソースを生成または受け入れることもできます。

前提条件

- OpenShift Container Platform のインストールおよび更新プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備を確認している。
- 「Agent-based Installer を使用したインストールの準備」ドキュメントを確認した。
- Red Hat Hybrid Cloud Console から Agent-based Installer とコマンドラインインターフェイス (CLI) をダウンロードしている。
- 管理者権限で OpenShift Container Platform にログインしている。

手順

1. 非接続環境の場合は、Red Hat OpenShift のミラーレジストリーをローカルコンテナーイメージレジストリーにミラーリングします。



重要

openshift-install バイナリーのバージョンが、Red Hat Quay などの共有レジストリーではなく、ローカルイメージコンテナーレジストリーに関連していることを確認してください。

```
$ ./openshift-install version
```

共有レジストリーバイナリーの出力例

```
./openshift-install 4.16.0
built from commit ae7977b7d1ca908674a0d45c5c243c766fa4b2ca
release image registry.ci.openshift.org/origin/release:4.16ocp-
release@sha256:0da6316466d60a3a4535d5fed3589feb0391989982fba59d47d
4c729912d6363
release architecture amd64
```

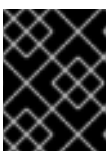
2. 組織のニーズに合わせて **install-config.yaml** 設定ファイルを設定します。

外部プラットフォームの設定を示す install-config.yaml 設定ファイルの例

```
# install-config.yaml
apiVersion: v1
baseDomain: <base_domain> (1)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  network type: OVNKubernetes
  machineNetwork:
  - cidr: <ip_address_from_cidr> (2)
  serviceNetwork:
  - 172.30.0.0/16
compute:
  - architecture: amd64 (3)
  hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  architecture: amd64 (4)
  hyperthreading: Enabled
  name: master
  replicas: 3
platform:
   external:
    platformName: oci (5)
    cloudControllerManager: External
sshKey: <public_ssh_key> (6)
pullSecret: '<pull_secret>' (7)
# ...
```

1. クラウドプロバイダーのベースドメイン。
2. ネットワーク上で機能するリソースおよびコンポーネントに対して CIDR が割り当てる仮想クラウドネットワーク (VCN) の IP アドレス。
3. 4. インフラストラクチャーに応じて、**x86_64** または **amd64** を選択できます。
5. OpenShift Container Platform が OCI と統合できるように、**OCI** を外部プラットフォームとして設定します。
6. SSH 公開鍵を指定します。
7. OpenShift Container Platform コンポーネントおよびサービス (Quay.io など) のコンテナーイメージをダウンロードするときに認証するために必要なプルシークレット。Red Hat Hybrid Cloud Console の Install OpenShift Container Platform 4 を参照してください。

3. ローカルシステム上に **openshift** という名前のディレクトリーを作成します。



重要

install-config.yaml および **agent-config.yaml** 設定ファイルを **openshift** ディレクトリーに移動しないでください。

4. **Oracle** ドキュメントの "Configuration Files" セクションの手順を完了して、Oracle Cloud Controller Manager (CCM) および Oracle Container Storage Interface (CSI) マニフェストをアーカイブファイルとしてダウンロードし、アーカイブファイルを **openshift** ディレクトリーに保存します。OpenShift Container Platform が外部 OCI プラットフォームに接続できるように、クラスターのインストール中に Oracle CCM をデプロイするために Oracle CCM マニフェストが必要です。OpenShift Container Platform が OCI から必要なオブジェクトを要求できるように、クラスターのインストール中に Oracle CSI ドライバーをデプロイするために Oracle CSI カスタムマニフェストが必要です。

5. **Oracle** ドキュメントの "Configuration Files" セクションで提供されているカスタムマニフェストファイルにアクセスします。

   a. **oci-ccm.yml** 設定ファイルで定義されている **oci-cloud-controller-manager** シークレットを、組織のリージョン、コンパートメント OCID、VCN OCID、およびロードバランサーのサブネット OCID と一致するように変更します。

6. OpenShift Container Platform CLI で次のコマンドを入力し、Agent-based Installer を使用して、rootfs イメージを除いた最小限の ISO イメージを生成します。後のプロセスでこのイメージを使用して、クラスターのすべてのノードを起動できます。

```
$ ./openshift-install agent create image --log-level debug
```

   このコマンドは次の操作も実行します。

   - サブディレクトリー **./<installation_directory>/auth directory:** を作成し、そのサブディレクトリーに **kubeadmin-password** ファイルと **kubeconfig** ファイルを配置します。
   - **agent-config.yaml** 設定ファイルで指定した IP アドレスに基づいて、 **rendezvousIP** ファイルを作成します。
   - オプション: **agent-config.yaml** および **install-config.yaml** 設定ファイルに加えた変更は、すべてゼロタッチプロビジョニング (ZTP) カスタムリソースにインポートされます。

重要

Agent-based Installer は Red Hat Enterprise Linux CoreOS (RHCOS) を使用します。後述のリスト項目で説明する rootfs イメージは、オペレーティングシステムの起動、復旧、修復に必要です。

7. 組織の要件に合わせて **agent-config.yaml** 設定ファイルを設定します。

IPv4 形式のネットワークの値を設定する agent-config.yaml 設定ファイルの例

```
apiVersion: v1alpha1
metadata:
  name: <cluster_name> 1
  namespace: <cluster_namespace> 2
rendezvousIP: <ip_address_from_CIDR> 3
bootArtifactsBaseURL: <server_URL> 4
# ...
```

1 DNS レコードに指定したクラスター名。

2 OpenShift Container Platform 上のクラスターの namespace。

3 ネットワーク IP アドレス形式として IPv4 を使用する場合は、**rendezvousIP** パラメーターを、VCN の Classless Inter-Domain Routing (CIDR) 方式によってネットワークに割り

4 rootfs イメージをアップロードするサーバーの URL。

8. 次のどちらかの更新を、**agent-config.yaml** 設定ファイルに適用します。

- 非接続ネットワークの場合: コマンドを実行して最小限の ISO イメージを生成すると、Agent-based Installer によって rootfs イメージがローカルシステムの **./<installation_directory>/boot-artifacts** ディレクトリーに保存されます。任意の Hypertext Transfer Protocol デーモン (**httpd**) などの優先 Web サーバーを使用して、**agent-config.yaml** 設定ファイルの **bootArtifactsBaseURL** パラメーターに指定された場所に rootfs をアップロードします。
  たとえば、**bootArtifactsBaseURL** パラメーターに **http://192.168.122.20** と指定されている場合、生成された rootfs イメージをこの場所にアップロードして、Agent-based Installer が **http://192.168.122.20/agent.x86_64-rootfs.img** からイメージにアクセスできるようにします。Agent-based Installer は、外部プラットフォームの最小限の ISO を起動した後、**http://192.168.122.20/agent.x86_64-rootfs.img** の場所から rootfs イメージをシステムメモリーにダウンロードします。

注記

また、Agent-based Installer は、Operator がクラスターのノードを起動するときに rootfs イメージをシステムメモリーにダウンロードするために、**bootArtifactsBaseURL** の値を最小限の ISO イメージの設定に追加します。

- 接続されたネットワークの場合: **agent-config.yaml** 設定ファイルで **bootArtifactsBaseURL** パラメーターを指定する必要はありません。Agent-based Installer のデフォルトの動作で、**https://rhcos.mirror.openshift.com** から rootfs URL の場所を読み取ります。Agent-based Installer は、外部プラットフォームの最小限の ISO を起動した後、デフォルトの RHCOS URL から rootfs ファイルをシステムのメモリーにダウンロードします。

重要

1 GB を超える完全な ISO イメージには、rootfs イメージが含まれていることに注意してください。このイメージは、通常 **150** MB 未満の最小 ISO イメージよりも大きくなります。

関連情報

- OpenShift Container Platform のインストール
- クラスターのインストールタイプの選択
- Agent-based Installer を使用したインストールの準備
- Agent-based Installer のダウンロード
- OpenShift Container Platform イメージリポジトリーのミラーリング
- オプション: ZTP マニフェストの使用

22.2.4. OpenShift Container Platform のファイアウォールの設定

OpenShift Container Platform をインストールする前に、ファイアウォールを、OpenShift Container Platform が必要とするサイトへのアクセスを付与するように設定する必要があります。ファイアウォールを使用する場合は、OpenShift Container Platform が機能するために必要なサイトにアクセスできるように、ファイアウォールに追加の設定を行います。

非接続環境の場合は、Red Hat と Oracle の両方のコンテンツをミラーリングする必要があります。このような環境では、ファイアウォールを特定のポートとレジストリーに公開するためのファイアウォールルールを作成する必要があります。

注記

ご使用の環境で OpenShift Container Platform クラスターの前に専用のロードバランサーがある場合は、ファイアウォールとロードバランサーの間の許可リストを確認して、クラスターに対する不要なネットワーク制限を回避してください。

手順

1. ファイアウォールの許可リストに次のレジストリー URL を設定します。

URL	ポート	機能
registry.redhat.io	443	コアコンテナーイメージを指定します。
access.redhat.com [1]	443	コアコンテナーイメージを含め、Red Hat Ecosytem Catalog に保存されているすべてのコンテナーイメージをホストします。
quay.io	443	コアコンテナーイメージを指定します。
cdn.quay.io	443	コアコンテナーイメージを指定します。
cdn01.quay.io	443	コアコンテナーイメージを指定します。
cdn02.quay.io	443	コアコンテナーイメージを指定します。
cdn03.quay.io	443	コアコンテナーイメージを指定します。
cdn04.quay.io	443	コアコンテナーイメージを指定します。
cdn05.quay.io	443	コアコンテナーイメージを指定します。
cdn06.quay.io	443	コアコンテナーイメージを指定します。
sso.redhat.com	443	**https://console.redhat.com** サイトは、**sso.redhat.com** からの認証を使用します。

1. ファイアウォール環境では、**access.redhat.com** リソースが許可リストに含まれていることを確認してください。このリソースは、コンテナークライアントが

registry.access.redhat.com からイメージを取得するときにイメージを検証するために必要な署名ストアをホストします。

許可リストで **cdn.quay.io** と **cdn0[1-3].quay.io** の代わりに、ワイルドカードの ***.quay.io** と ***.openshiftapps.com** を使用できます。**quay.io** などのサイトを許可リストに追加するには、***.quay.io** などのワイルドカードエントリーを拒否リストに加えないでください。ほとんどの場合、イメージレジストリーはコンテンツ配信ネットワーク (CDN) を使用してイメージを提供します。ファイアウォールがアクセスをブロックすると、最初のダウンロード要求が **cdn01.quay.io** などのホスト名にリダイレクトされるときに、イメージのダウンロードが拒否されます。

2. ファイアウォールの許可リストを設定し、ビルドに必要な言語またはフレームワークのリソースを提供するサイトをリストに含めます。

3. Telemetry を無効にしていない場合は、以下の URL へのアクセスを許可して Red Hat Insights にアクセスできるようにする必要があります。

URL	ポート	機能
cert-api.access.redhat.com	443	Telemetry で必須
api.access.redhat.com	443	Telemetry で必須
infogw.api.openshift.com	443	Telemetry で必須
console.redhat.com	443	Telemetry および **insights-operator** で必須

4. ファイアウォールの許可リストを設定し、次のレジストリー URL をリストに含めます。

URL	ポート	機能
api.openshift.com	443	クラスタートークンの両方が必要であり、クラスターに更新が利用可能かどうかを確認するために必要です。
rhcos.mirror.openshift.com	443	Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードするために必要。

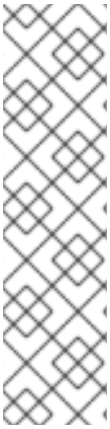
5. ファイアウォールの許可リストを設定し、次の外部 URL をリストに含めます。各リポジトリー URL は OCI コンテナーをホストします。パフォーマンスの問題を軽減するために、イメージをミラーリングするリポジトリーの数をできる限り少なくすることを検討してください。

URL	ポート	機能

URL	ポート	機能
k8s.gcr.io	port	コミュニティーベースのイメージレジストリーのコンテナーイメージをホストする Kubernetes レジストリー。このイメージレジストリーは、カスタム Google Container Registry (GCR) ドメインでホストされています。
ghcr.io	port	Open Container Initiative イメージを保存および管理できる GitHub イメージレジストリー。プライベート、内部、パブリックパッケージを公開、インストール、削除するには、アクセストークンが必要です。
storage.googleapis.com	443	リリースイメージ署名のソース (ただし、Cluster Version Operator には単一の機能ソースのみが必要)。
registry.k8s.io	port	**k8s.gcr.io** イメージレジストリーを置き換えます。これは、**k8s.gcr.io** イメージレジストリーが他のプラットフォームやベンダーをサポートしていないためです。

22.2.5. OCI でのクラスターの実行

Oracle® Cloud Infrastructure (OCI) でクラスターを実行するには、生成されたエージェント ISO イメージを OCI のデフォルトの Object Storage バケットにアップロードする必要があります。さらに、OCI 上でクラスターを実行するために OpenShift Container Platform と OCI が相互に通信できるように、提供されたベースイメージからコンピュートインスタンスを作成する必要があります。



注記

OCI は、次の OpenShift Container Platform クラスタートポロジーをサポートしています。

- シングルノードへの OpenShift Container Platform クラスターのインストール
- 少なくとも 3 つのコントロールプレーンインスタンスと 2 つのコンピュートインスタンスを持つ高可用性クラスター
- 少なくとも 3 つのコントロールプレーンインスタンスを持つコンパクトな 3 ノードクラスター

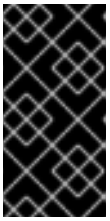
前提条件

- エージェント ISO イメージを生成した。「OCI にクラスターをインストールするための設定ファイルの作成」を参照してください。

手順

1. エージェント ISO イメージを Oracle のデフォルトの Object Storage バケットにアップロードし、エージェント ISO イメージをカスタムイメージとしてこのバケットにインポートします。カスタムイメージが Unified Extensible Firmware Interface (UEFI) モードで起動するように設定されていることを確認します。詳細は、Creating the OpenShift Container Platform ISO Image (Oracle ドキュメント) を参照してください。

2. クラスタートポロジー用に提供されたベースイメージからコンピュートインスタンスを作成します。Creating the OpenShift Container Platform cluster on OCI (Oracle ドキュメント) を参照してください。



重要

コンピュートインスタンスを作成する前に、クラスターに十分なメモリーとディスクリソースがあることを確認してください。さらに、少なくとも 1 つのコンピュートインスタンスが、**agent-config.yaml** ファイルの **rendezvousIP** に記載されているアドレスと同じ IP アドレスを持っていることを確認してください。

関連情報

- 各トポロジーに推奨されるリソース
- Instance Sizing Recommendations for OpenShift Container Platform on OCI Nodes (Oracle ドキュメント)
- Troubleshooting OpenShift Container Platform on OCI (Oracle ドキュメント)

22.2.6. エージェントベースのクラスターのインストールが OCI 上で実行されることを確認する

クラスターがインストールされ、Oracle® Cloud Infrastructure (OCI) 上で効果的に実行されていることを確認します。

前提条件

- 必要な OCI リソースとサービスをすべて作成した。「OCI インフラストラクチャーのリソースとサービスの作成」を参照してください。
- **install-config.yaml** 設定ファイルと **agent-config.yaml** 設定ファイルを作成した。「OCI にクラスターをインストールするための設定ファイルの作成」を参照してください。
- エージェント ISO イメージを Oracle のデフォルトの Object Storage バケットにアップロードし、OCI 上にコンピュートインスタンスを作成した。詳細は、「OCI でのクラスターの実行」を参照してください。

手順

OpenShift Container Platform クラスター内の自己管理ノードにコンピュートインスタンスをデプロイした後、以下のいずれかの方法でクラスターのステータスを監視できます。

- OpenShift Container Platform CLI から次のコマンドを入力します。

```
$ ./openshift-install agent wait-for install-complete --log-level debug
```

ブートストラップノードが動作する **rendezvous** ホストノードの状態を確認します。ホストが再起動すると、ホストはクラスターの一部となります。

- **kubeconfig** API を使用して、さまざまな OpenShift Container Platform コンポーネントのステータスを確認します。**KUBECONFIG** 環境変数に、クラスターの **kubeconfig** 設定ファイルの相対パスを設定します。

```
$  export KUBECONFIG=~/auth/kubeconfig
```

クラスターの各自己管理ノードのステータスを確認します。CCM は、各ノードにラベルを適用して、ノードを OCI 上のクラスターで実行するよう指定します。

```
$ oc get nodes -A
```

出力例

```
NAME                                  STATUS ROLES                AGE VERSION
main-0.private.agenttest.oraclevcn.com Ready  control-plane, master 7m  v1.27.4+6eeca63
main-1.private.agenttest.oraclevcn.com Ready  control-plane, master 15m v1.27.4+d7fa83f
main-2.private.agenttest.oraclevcn.com Ready  control-plane, master 15m v1.27.4+d7fa83f
```

クラスターの各 Operator のステータスを確認します。CCM Operator のステータスは、クラスターが実行中であることを示す適切な指標です。

```
$ oc get co
```

出力例 (一部のみ記載)

```
NAME          VERSION     AVAILABLE  PROGRESSING    DEGRADED   SINCE
MESSAGE
authentication 4.16.0-0    True       False          False      6m18s
baremetal      4.16.0-0    True       False          False      2m42s
network        4.16.0-0    True       True           False      5m58s  Progressing: …
    …
```

関連情報

- 失敗したエージェントベースのインストールからログデータを収集する

第23章 VSPHERE へのインストール

23.1. インストール方法

さまざまなインストール方法を使用して、OpenShift Container Platform クラスターを vSphere にインストールできます。それぞれの方法は、その特質上、適しているユースケース (非接続環境にクラスターをインストールする場合や、最小限の設定とプロビジョニングでクラスターをインストールする場合など) が異なります。

23.1.1. Assisted Installer

Assisted Installer を使用して、OpenShift Container Platform をインストールできます。この方法はインストーラーのセットアップが不要で、vSphere などの接続環境に最適です。Assisted Installer を使用してインストールすると、vSphere との統合も提供され、自動スケーリングが可能になります。詳細は、自動インストーラーを使用したオンプレミスクラスターのインストール を参照してください。

23.1.2. Agent-based Installer

Agent-based Installer を使用して、OpenShift Container Platform クラスターを vSphere にインストールできます。Agent-based Installer を使用すると、起動可能なイメージを使用して、非接続環境でオンプレミスサーバーを起動できます。Agent-based Installer を使用すると、ユーザーはインフラストラクチャーのプロビジョニング、ネットワーク設定のカスタマイズ、非接続環境内でのインストールのカスタマイズを柔軟に行うことができます。詳細は、Agent-based Installer を使用したインストールの準備 を参照してください。

23.1.3. installer-provisioned infrastructure によるインストール

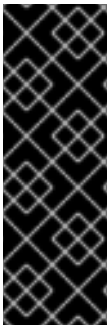
installer-provisioned infrastructure を使用して、OpenShift Container Platform を vSphere にインストールできます。installer-provisioned infrastructure により、インストールプログラムは OpenShift Container Platform で必要なリソースのプロビジョニングを事前に設定し、自動化することができます。installer-provisioned infrastructure は、ネットワーク非接続環境にインストールする場合に役立ちます。このような環境では、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。

- **クラスターの vSphere へのインストール**: インストーラーでプロビジョニングされるインフラストラクチャーのインストールをカスタマイズせずに使用して、vSphere に OpenShift Container Platform をインストールできます。
- **カスタマイズによる vSphere へのクラスターのインストール**: インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトのカスタマイズオプションのインストールを使用して、vSphere に OpenShift Container Platform をインストールできます。
- **ネットワークのカスタマイズによる vSphere へのクラスターのインストール**: ネットワークのカスタマイズを使用して、インストーラーでプロビジョニングされる vSphere インフラストラクチャーに OpenShift Container Platform をインストールできます。インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。
- **ネットワークが制限された環境での vSphere へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを作成して、ネットワークが制限された環境で VMware vSphere インフラストラクチャーにクラスターをインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

23.1.4. user-provisioned infrastructure によるインストール

user-provisioned infrastructure を使用して、OpenShift Container Platform を vSphere にインストールできます。ユーザーによってプロビジョニングされるインフラストラクチャーでは、ユーザーは OpenShift Container Platform に必要なすべてのリソースをプロビジョニングする必要があります。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

- **user-provisioned infrastructure での vSphere へのクラスターのインストール**: 独自にプロビジョニングする VMware vSphere インフラストラクチャーに OpenShift Container Platform をインストールできます。
- **カスタマイズされたネットワークを使用したユーザーによってプロビジョニングされるインフラストラクチャーの vSphere へのクラスターのインストール**: カスタマイズされたネットワーク設定オプションを使用して独自にプロビジョニングする VMware vSphere インフラストラクチャーに OpenShift Container Platform をインストールできます。
- **ネットワークが制限された環境でユーザーによってプロビジョニングされるインフラストラクチャーの vSphere へのクラスターのインストール**: ネットワークが制限された環境でプロビジョニングされる VMware vSphere インフラストラクチャーに、OpenShift Container Platform をインストールできます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

23.1.5. 関連情報

- インストールプロセス

23.2. INSTALLER-PROVISIONED INFRASTRUCTURE

23.2.1. vSphere のインストール要件

installer-provisioned infrastructure を使用してインストールを開始する前に、vSphere 環境が次のインストール要件を満たしていることを確認してください。

23.2.1.1. VMware vSphere インフラストラクチャーの要件

OpenShift Container Platform クラスターは、使用するコンポーネントの要件に合わせて、以下に示す VMware vSphere インスタンスのいずれかのバージョンにインストールする必要があります。

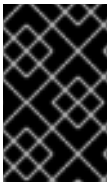
- バージョン 7.0 Update 2 以降
- バージョン 8.0 Update 1 以降

これらのリリースは、どちらも Container Storage Interface (CSI) の移行をサポートしています。CSI の移行は、OpenShift Container Platform 4.16 ではデフォルトで有効になっています。

VMware vSphere インフラストラクチャーは、オンプレミスまたは次の表に示す要件を満たす VMware Cloud Verified プロバイダー でホストできます。

表23.1 vSphere 仮想環境のバージョン要件

仮想環境製品	必須バージョン
VMware 仮想ハードウェア	15 以降
vSphere ESXi ホスト	7.0 Update 2 以降、8.0 Update 1 以降
vCenter ホスト	7.0 Update 2 以降、8.0 Update 1 以降



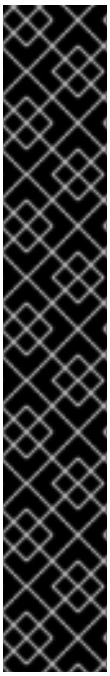
重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

表23.2 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	仮想ハードウェアバージョン 15 を搭載した vSphere 7.0 Update 2 (以降) または vSphere 8.0 Update 1 (以降)	このハイパーバイザーのバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。RHCOS と互換性のある Red Hat Enterprise Linux (RHEL) の最新バージョンでサポートされているハードウェアの詳細は、Red Hat Customer Portal の ハードウェア を参照してください。
オプション: Networking(NSX-T)	vSphere 7.0 Update 2 以降、vSphere 8.0 Update 1 以降	OpenShift Container Platform には、vSphere 7.0 Update 2 以降または vSphere 8.0 Update 1 以降が必要です。NSX および OpenShift Container Platform の互換性についての詳細は、VMware の NSX コンテナープラグインドキュメント の リリースノート セクションを参照してください。

コンポーネント	サポートされる最小バージョン	説明
CPU マイクロアーキテクチャー	x86-64-v2 以降	OpenShift 4.13 以降では、マイクロアーキテクチャーの要件が x86-64-v2 に発生する RHEL 9.2 ホストオペレーティングシステムをベースにしています。RHEL マイクロアーキテクチャー要件 に関するドキュメント を参照してください。このナレッジベースの記事 に記載されている手順に従って、互換性を確認できます。



重要

Oracle® Cloud Infrastructure (OCI) および Oracle® Cloud VMware Solution (OCVS) サービス上で動作するクラスターワークロードの最適なパフォーマンス条件を確保するには、ブロックボリュームのボリュームパフォーマンスユニット (VPU) がワークロードに合わせてサイズ設定されていることを確認してください。

次のリストを参考に、特定のパフォーマンスニーズに応じて必要な VPU を選択してください。

- テストまたは概念実証環境: 100 GB、20 - 30 VPU。
- 基本実稼働環境: 500 GB、60 VPU。
- 頻繁に使用される実稼働環境: 500 GB 以上、100 以上の VPU。

更新とスケーリングアクティビティーに十分な容量を提供するために、追加の VPU を割り当てることを検討してください。Block Volume Performance Levels (Oracle ドキュメント) を参照してください。

23.2.1.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表23.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
VRRP	該当なし	keepalived に必要
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック

プロトコル	ポート	説明
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表23.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表23.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

23.2.1.3. VMware vSphere CSI Driver Operator の要件

vSphere Container Storage Interface (CSI) Driver Operator をインストールするには、次の要件を満たす必要があります。

- VMware vSphere バージョン: 7.0 Update 2 以降、8.0 Update 1 以降
- vCenter バージョン: 7.0 Update 2 以降、8.0 Update 1 以降
- ハードウェアバージョン 15 以降の仮想マシン

- クラスターにサードパーティーの vSphere CSI ドライバーがインストールされていない

サードパーティーの vSphere CSI ドライバーがクラスターに存在する場合、OpenShift Container Platform はそれを上書きしません。サードパーティーの vSphere CSI ドライバーが存在すると、OpenShift Container Platform を OpenShift Container Platform 4.13 以降にアップグレードできなくなります。

注記

VMware vSphere CSI Driver Operator は、インストールマニフェストの **platform: vsphere** でデプロイされたクラスターでのみサポートされます。

Container Storage Interface (CSI) ドライバー、vSphere CSI Driver Operator、および vSphere Problem Detector Operator のカスタムロールを作成できます。カスタムロールには、各 vSphere オブジェクトに最小限の権限セットを割り当てる権限セットを含めることができます。つまり、CSI ドライバー、vSphere CSI Driver Operator、および vSphere Problem Detector Operator はこれらのオブジェクトとの基本的な対話を確立できます。

重要

vCenter への OpenShift Container Platform クラスターのインストールは、「必要な vCenter アカウントの特権」セクションで説明されているすべての特権のリストに対してテストされています。このすべての特権のリストに準拠することで、制限された特権セットを持つカスタムロールの作成時に予期しない動作やサポートされていない動作が発生する可能性を抑制できます。

関連情報

- サードパーティーの vSphere CSI ドライバーを削除する場合は、サードパーティーの vSphere CSI ドライバーの削除 を参照してください。
- vSphere ノードのハードウェアバージョンを更新する場合は、vSphere で稼働するノードのハードウェア更新 を参照してください。
- ストレージコンポーネントの最小権限

23.2.1.4. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、これを修正する必要があります。

インストールフロクラムが vSphere 仮想マシンフォルターを作成するために使用される場合には、追加のロールが必要です。

例23.1 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	常時	**Cns.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.Update** **StorageProfile.View**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	常時	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**
vSphere ポートグループ	常時	**Network.Assign**
仮想マシンフォルダー	常時	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.Add**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		NewDisk **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.EditDevice** **VirtualMachine.Config.Memory** **VirtualMachine.Config.RemoveDisk** **VirtualMachine.Config.Rename** **Host.Config.Storage** **VirtualMachine.Config.ResetGuestInfo** **VirtualMachine.Config.Resource** **VirtualMachine.Config.Settings** **VirtualMachine.Config.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.MarkAsTemplate** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラス	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VirtualMachine.Config.Add**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
	ターで Machine API を使用しない場合は、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。「Machine API の最小権限」の表を参照してください。	**ExistingDisk** **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.EditDevice** **VirtualMachine.Config.Memory** **VirtualMachine.Config.RemoveDisk** **VirtualMachine.Config.Rename** **VirtualMachine.Config.ResetGuestInfo** **VirtualMachine.Config.Resource** **VirtualMachine.Config.Settings** **VirtualMachine.Config.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.DeployTemplate** **VirtualMachine.Provisioning.MarkAsTemplate** **Folder.Create** **Folder.Delete**

例23.2 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	常時	**Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere Datastore	常時	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	常時	**Network."Assign network"**
仮想マシンフォルダー	常時	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。	"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		template" Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

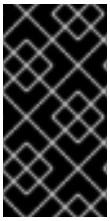
例23.3 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	必要になる場合	子への伝播	パーミッションが必要
vSphere vCenter	常時	False	必要な特権がリスト表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権がリスト表示
vSphere vCenter Cluster	既存のリソースプール	False	**ReadOnly** パーミッション
	クラスタールートの仮想マシン	True	必要な特権がリスト表示
vSphere vCenter Datastore	常時	False	必要な特権がリスト表示
vSphere Switch	常時	False	**ReadOnly** パーミッション
vSphere ポートグループ	常時	False	必要な特権がリスト表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権がリスト表示
vSphere vCenter リソースプール	既存のリソースプール	True	必要な特権がリスト表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

最低限必要な vCenter アカウントの特権
カスタムロールを作成してそのロールに特権を割り当てた後、特定の vSphere オブジェクトを選択し、オブジェクトごとにカスタムロールをユーザーまたはグループに割り当てることで権限を作成できます。

vSphere オブジェクトの権限を作成するか、権限の作成を要求する前に、vSphere オブジェクトに適用する最小限の権限を決定してください。このタスクを実行すると、vSphere オブジェクトと OpenShift Container Platform アーキテクチャーの間の基本的な対話を確立できます。



重要

カスタムロールを作成し、そのロールに特権を割り当てなかった場合、vSphere Server はデフォルトで **Read Only** ロールをそのカスタムロールに割り当てます。クラウドプロバイダー API の場合、カスタムロールは **Read Only** ロールの特権を継承するだけで済むことに注意してください。

グローバル管理者の特権を持つアカウントがニーズに合わない場合は、カスタムロールの作成を検討してください。

重要

必要な特権が設定されていないアカウントはサポートされません。vCenter への OpenShift Container Platform クラスターのインストールは、「必要な vCenter アカウントの特権」セクションで説明されているすべての特権のリストに対してテストされています。このすべての権限のリストに準拠することで、制限された特権セットを持つカスタムロールの作成時に予期しない動作が発生する可能性を抑制できます。

以下の表に、特定の OpenShift Container Platform アーキテクチャーと対話する vSphere オブジェクトの最小権限のリストを示します。

例23.4 installer-provisioned infrastructure の最小権限

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	常時	**Cns.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.Update StorageProfile.View**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter Cluster	クラスタールートに仮想マシンを作成する場合	**Host.Config.Storage Resource.AssignVMT oPool VApp.AssignResourc ePool VApp.Import VirtualMachine.Config .AddNewDisk**
vSphere vCenter リソースプール	**install-config.yaml** ファイルに既存のリソースプールを指定する場合	**Datastore.Browse Datastore.FileManage ment Host.Config.Storage InventoryService.Tag ging.ObjectAttachable Resource.AssignVMT oPool VApp.AssignResourc ePool VApp.Import`minimu m**
vSphere ポートグループ	常時	**Network.Assign**
仮想マシンフォルダー	常時	**InventoryService.Tag ging.ObjectAttachable Resource.AssignVMT oPool VApp.Import VirtualMachine.Config .AddExistingDisk VirtualMachine.Config .AddNewDisk VirtualMachine.Config .AddRemoveDevice VirtualMachine.Config .AdvancedConfig VirtualMachine.Config .Annotation VirtualMachine.Config .CPUCount VirtualMachine.Config .DiskExtend VirtualMachine.Config .DiskLease VirtualMachine.Config .EditDevice VirtualMachine.Config .Memory VirtualMachine.Config .RemoveDisk VirtualMachine.Config**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
		.Rename **VirtualMachine.Config** **.ResetGuestInfo** **VirtualMachine.Config** **.Resource** **VirtualMachine.Config** **.Settings** **VirtualMachine.Config** **.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.MarkAsTemplate** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。クラスターで Machine API を使用しており、API の最小権限セットを設定する必要がある場合は、「Machine API の最小権限」の表を参照してください。	**Folder.Create** **Folder.Delete** **InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config** **.AddExistingDisk** **VirtualMachine.Config** **.AddNewDisk** **VirtualMachine.Config** **.AddRemoveDevice** **VirtualMachine.Config** **.AdvancedConfig** **VirtualMachine.Config** **.Annotation** **VirtualMachine.Config** **.CPUCount** **VirtualMachine.Config** **.DiskExtend** **VirtualMachine.Config**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
		.DiskLease VirtualMachine.Config .EditDevice VirtualMachine.Config .Memory VirtualMachine.Config .RemoveDisk VirtualMachine.Config .Rename VirtualMachine.Config .ResetGuestInfo VirtualMachine.Config .Resource VirtualMachine.Config .Settings VirtualMachine.Config .UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate

例23.5 コンポーネントのインストール後の管理のための最小権限

ロールの vSphere オブジェクト	必要になる場合	必要な特権

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	常時	**Cns.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.Update StorageProfile.View**
vSphere vCenter Cluster	クラスタールートに仮想マシンを作成する場合	**Host.Config.Storage Resource.AssignVMToPool**
vSphere vCenter リソースプール	**install-config.yaml** ファイルに既存のリソースプールを指定する場合	**Host.Config.Storage**
vSphere Datastore	常時	**Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement InventoryService.Tagging.ObjectAttachable**
vSphere ポートグループ	常時	**Network.Assign**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
仮想マシンフォルダー	常時	**VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.Memory** **VirtualMachine.Config.Settings** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。クラスターで Machine API を使用しており、API の最小権限セットを設定する必要がある場合は、「Machine API の最小権限」の表を参照してください。	**Resource.AssignVMToPool** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Provisioning.DeployTemplate**

例23.6 ストレージコンポーネントの最小権限

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	常時	**Cns.Searchable** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **StorageProfile.Update** **StorageProfile.View**
vSphere vCenter Cluster	クラスタールートに仮想マシンを作成する場合	**Host.Config.Storage**
vSphere vCenter リソースプール	**install-config.yaml** ファイルに既存のリソースプールを指定する場合	**Host.Config.Storage**
vSphere Datastore	常時	**Datastore.Browse** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**
vSphere ポートグループ	常時	**Read-Only**
仮想マシンフォルダー	常時	**VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddRemoveDevice**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。クラスターで Machine API を使用しており、API の最小権限セットを設定する必要がある場合は、「Machine API の最小権限」の表を参照してください。	**VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddRemoveDevice**

例23.7 Machine API の最小権限

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	常時	**InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.Update StorageProfile.View**
vSphere vCenter Cluster	クラスタールートに仮想マシンを作成する場合	**Resource.AssignVMToPool**
vSphere vCenter リソースプール	**install-config.yaml** ファイルに既存のリソースプールを指定する場合	**Read-Only**
vSphere Datastore	常時	**Datastore.AllocateSpace Datastore.Browse**
vSphere ポートグループ	常時	**Network.Assign**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
仮想マシンフォルダー	常時	**VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.Memory** **VirtualMachine.Config.Settings** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。	**Resource.AssignVMToPool** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Provisioning.DeployTemplate**

OpenShift Container Platform と vMotion の使用

vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュート専用の vMotion をサポートします。これは、**一般に**、vMotion に関するすべての VMware ベストプラクティスを満たすことを意味します。
  コンピュートプレーンノードとコントロールプレーンノードの稼働時間を確保するには、vMotion に関する VMware のベストプラクティスに従い、VMware のアンチアフィニティールールを使用して、メンテナンスまたはハードウェアの問題時の OpenShift Container Platform の可用性を向上させます。

vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。Pod で vSphere ボリュームを使用している場合、手動または Storage vMotion を介してデータストア間で VM を移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生し、データ損失が発生する可能性があります。

- OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

重要

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion を使用する Storage Distributed Resource Scheduler (SDRS) がデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage DRS を無効にする必要があります。

複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

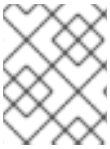
- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに動的ホスト設定プロトコル (DHCP) を使用し、クラスターマシンに永続的な IP アドレスを提供するように DHCP サーバーが設定されていることを確認します。



注記

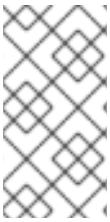
静的 IP アドレスを使用してノードをプロビジョニングする場合は、ネットワークに DHCP を使用する必要はありません。

DHCP サーバーを使用するには、デフォルトゲートウェイを設定します。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。

ネットワークに動的ホスト設定プロトコル (DHCP) を使用し、クラスターマシンに永続的な IP アドレスを提供するように DHCP サーバーが設定されていることを確認する必要があります。DHCP リースでは、デフォルトゲートウェイを使用するように DHCP を設定する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。

制限された環境にインストールする場合、制限されたネットワーク内の仮想マシンは、ノード、永続ボリュームクレーム (PVC)、およびその他のリソースをプロビジョニングおよび管理するために、vCenter にアクセスできる必要があります。

さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。



注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
DHCP を使用するネットワークの場合、インストーラによってプロビジョニングされる vSphere インストールには 2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するク

ラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表23.6 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME (Canonical Name) レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

vSphere ノードの静的 IP アドレス
動的ホスト設定プロトコル (DHCP) が存在しない環境では、ブートストラップ、コントロールプレーン、およびコンピュートノードを静的 IP アドレスで設定するようにプロビジョニングできます。この環境を設定するには、**install-config.yaml** ファイルの **platform.vsphere.hosts.role** パラメーターに値を指定する必要があります。

デフォルトでは、インストールプログラムはネットワークに DHCP を使用するように設定されていますが、このネットワークの設定可能な機能は限られています。

install-config.yaml ファイルで 1 つ以上のマシンプールを定義した後、ネットワーク上のノードにネットワーク定義を定義できます。ネットワーク定義の数が、クラスターに設定したマシンプールの数と一致していることを確認してください。

次の例は、**compute** ロールを持つノードのネットワーク設定を示しています。

```
---
platform:
  vsphere:
    hosts:
    - role: compute 1
      networkDevice:
        ipAddrs:
        - 192.168.204.10/24 2
        gateway: 192.168.204.1 3
```

```
      nameservers:
      - 192.168.204.1 4
---
```

1 有効なネットワーク定義値には、**bootstrap**、**control-plane**、および **compute** が含まれます。**install-config.yaml** 設定ファイルに少なくとも 1 つの **bootstrap** ネットワーク定義をリストする必要があります。

2 インストールプログラムがネットワークインターフェイスに渡す IPv4、IPv6、またはその両方の IP アドレスをリストします。マシン API コントローラーは、設定されているすべての IP アドレスをデフォルトのネットワークインターフェイスに割り当てます。

3 ネットワークインターフェイスのデフォルトゲートウェイ。

4 最大 3 つの DNS ネームサーバーをリストします。

静的 IP アドレスを持つノードを実行するようにクラスターをデプロイした後、これらの静的 IP アドレスのいずれかを使用するようにマシンをスケーリングできます。さらに、マシンセットを使用して、設定済みの静的 IP アドレスの 1 つを使用するようにマシンを設定できます。

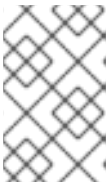
関連情報

- 静的 IP アドレスを使用するようにマシンをスケーリングする
- マシンセットを使用して設定された静的 IP アドレスを持つマシンをスケールする

23.2.2. installer-provisioned infrastructure を使用したクラスターのインストールの準備

以下の手順を実行して、vSphere に OpenShift Container Platform クラスターをインストールする準備をします。

- インストールプログラムをダウンロードします。



注記

非接続環境にインストールする場合は、ミラーリングしたコンテンツからインストールプログラムを抽出します。詳細は、非接続インストール用のイメージのミラーリング を参照してください。

- OpenShift CLI (**oc**) をインストールします。

注記

非接続環境にインストールする場合は、ミラーホストに **oc** をインストールします。

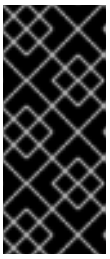
- SSH キーペアを生成します。OpenShift Container Platform クラスターのデプロイ後にこのキーペアを使用して、クラスターのノードに対する認証を行うことができます。
- vCenter の信頼されたルート CA 証明書をシステム信頼に追加します。

23.2.2.1. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

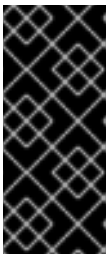


重要

macOS 上でインストールプログラムを実行しようとすると、**golang** コンパイラーに関連する既知の問題により、OpenShift Container Platform クラスターのインストールに失敗します。この問題の詳細は、**OpenShift Container Platform 4.16 リリースノート** ドキュメントの「既知の問題」セクションを参照してください。

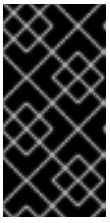
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

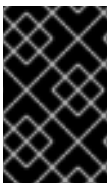
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

23.2.2.2. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

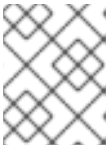
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

23.2.2.3. クラスターノードの SSH アクセス用のキーペアの生成

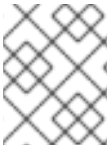
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

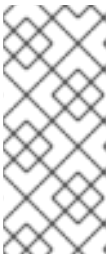
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

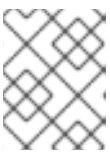
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

23.2.2.4. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
```

```
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

23.2.3. クラスターの vSphere へのインストール

OpenShift Container Platform バージョン 4.16 では、installer-provisioned infrastructure を使用して、VMware vSphere インスタンスにクラスターをインストールできます。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

23.2.3.1. 前提条件

- installer-provisioned infrastructure を使用したクラスターのインストールの準備 のタスクを完了した。
- VMware プラットフォームのライセンスを確認した。Red Hat は VMware ライセンスに制限を設けていませんが、一部の VMware インフラストラクチャーコンポーネントにはライセンスが必要です。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターの 永続ストレージ をプロビジョニングした。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。

- OpenShift Container Platform インストーラーは、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

23.2.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

23.2.3.3. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

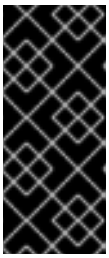
重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。
- オプション: クラスターを作成する前に、デフォルトのロードバランサーの代わりに外部ロードバランサーを設定します。



重要

インストールプログラムに API および Ingress 静的アドレスを指定する必要はありません。この設定を選択した場合は、追加のアクションを実行して、参照される各 vSphere サブネットから IP アドレスを受け入れるネットワークターゲットを定義する必要があります。「ユーザー管理ロードバランサーの設定」セクションを参照してください。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

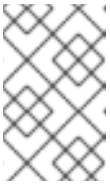
1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

ディレクトリーを指定する場合:

- ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
- 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. プロンプト時に値を指定します。
   a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



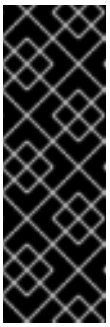
注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

   b. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

c. vCenter インスタンスの名前を指定します。

d. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
インストールプログラムは vCenter インスタンスに接続します。



重要

Active Directory (AD) が統合された一部の VMware vCenter Single Sign-On (SSO) 環境では、主に **<domain>** 構造を必要とする従来のログイン方法を使用する必要がある可能性があります。

vCenter アカウントの権限チェックが必ず適切に完了するようにするには、**<username>@<full_qualified_domainname>** などのユーザープリンシパル名 (UPN) ログイン方法の使用を検討してください。

e. 接続する vCenter インスタンスのデータセンターを選択します。

f. 使用するデフォルトの vCenter データストアを選択します。

注記

データストアとクラスター名は 60 文字を超えることができません。そのため、組み合わせた文字列の長さが 60 文字の制限を超えないようにしてください。

g. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

h. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

i. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

j. クラスター Ingress に設定した仮想 IP アドレスを入力します。

k. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

l. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

注記

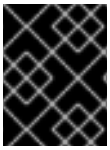
データストアとクラスター名は 60 文字を超えることができません。そのため、組み合わせた文字列の長さが 60 文字の制限を超えないようにしてください。

m. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

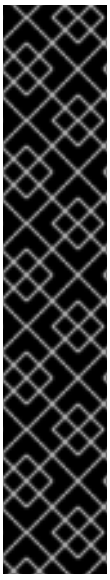


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

23.2.3.4. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

23.2.3.5. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

23.2.3.5.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

23.2.3.5.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

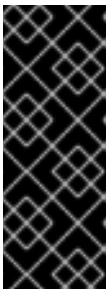
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

23.2.3.5.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Data Foundation など、クラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- "100Gi" の容量が必要です。

重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリックストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

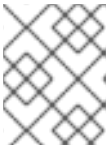
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

23.2.3.5.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、**1** つのレプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
```

```
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
4. 永続ボリューム要求 (PVC) のサイズ。

b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1. カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

正しい PVC を参照するようにレジストリーストレージを設定する手順は、vSphere のレジストリーの設定 を参照してください。

23.2.3.6. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

23.2.3.7. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

23.2.4. カスタマイズによる vSphere へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、installer-provisioned infrastructure を使用して、VMware vSphere インスタンスにクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

23.2.4.1. 前提条件

- installer-provisioned infrastructure を使用したクラスターのインストールの準備 のタスクを完了した。
- VMware プラットフォームのライセンスを確認した。Red Hat は VMware ライセンスに制限を設けていませんが、一部の VMware インフラストラクチャーコンポーネントにはライセンスが必要です。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターの 永続ストレージ をプロビジョニングした。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform インストーラーは、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

23.2.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

23.2.4.3. VMware vSphere のリージョンとゾーンの有効化

OpenShift Container Platform クラスターを、単一の VMware vCenter で実行される複数の vSphere データセンターにデプロイできます。各データセンターは複数のクラスターを実行できます。この設定により、クラスターの障害を引き起こす可能性のあるハードウェア障害やネットワーク停止のリスクが軽減されます。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

重要

VMware vSphere のリージョンおよびゾーンの有効化機能には、クラスター内のデフォルトのストレージドライバーとして vSphere Container Storage Interface (CSI) ドライバーが必要です。そのため、この機能は新しくインストールされたクラスターでのみ使用できます。

以前のリリースからアップグレードされたクラスターの場合は、クラスターの CSI 自動移行を有効にする必要があります。その後、アップグレードされたクラスターに対して複数のリージョンとゾーンを設定できます。

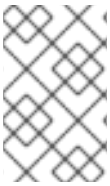
デフォルトのインストール設定では、クラスターが単一の vSphere データセンターにデプロイされます。クラスターを複数の vSphere データセンターにデプロイする場合は、リージョンおよびゾーン機能を有効にするインストール設定ファイルを作成する必要があります。

デフォルトの **install-config.yaml** ファイルには **vcenters フィールド** と **FailureDomains** フィールドが含まれており、OpenShift Container Platform クラスターに複数の vSphere データセンターとクラスターを指定できます。単一のデータセンターで設定される vSphere 環境に OpenShift Container

Platform クラスターをインストールする場合は、これらのフィールドを空白のままにすることができます。

次のリストでは、クラスターのゾーンとリージョンの定義に関連する用語について説明します。

- 障害ドメイン: リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。
- リージョン: vCenter データセンターを指定します。リージョンを定義するには、**openshift-region** タグカテゴリーのタグを使用します。
- ゾーン: vCenter クラスターを指定します。ゾーンを定義するには、**openshift-zone** タグカテゴリーのタグを使用します。



注記

install-config.yaml ファイルで複数の障害ドメインを指定する予定がある場合は、設定ファイルを作成する前に、タグカテゴリー、ゾーンタグ、およびリージョンタグを作成する必要があります。

リージョンを表す vCenter データセンターごとに vCenter タグを作成する必要があります。さらに、データセンターで実行されるクラスターごとに、ゾーンを表す vCenter タグを作成する必要があります。タグを作成した後、各タグをそれぞれのデータセンターとクラスターにアタッチする必要があります。

次の表は、単一の VMware vCenter で実行されている複数の vSphere データセンターを含む設定のリージョン、ゾーン、タグ間の関係の例を示しています。

データセンター (リージョン)	クラスター (ゾーン)	タグ
米国東部	us-east-1	us-east-1a
		us-east-1b
	us-east-2	us-east-2a
		us-east-2b
us-west	us-west-1	us-west-1a
		us-west-1b
	us-west-2	us-west-2a
		us-west-2b

関連情報

- 追加の VMware vSphere 設定パラメーター

- 非推奨の VMware vSphere 設定パラメーター
- vSphere の自動移行
- VMware vSphere CSI ドライバー Operator

23.2.4.4. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。
      iii. vCenter インスタンスの名前を指定します。

iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
インストールプログラムは vCenter インスタンスに接続します。

v. 接続する vCenter インスタンスのデータセンターを選択します。

注記

インストール設定ファイルを作成した後、そのファイルを変更して複数の vSphere データセンター環境を作成できます。これは、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできることを意味します。この環境の作成の詳細については、**VMware vSphere のリージョンとゾーンの有効化** を参照してください。

vi. 使用するデフォルトの vCenter データストアを選択します。

警告

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion を使用する Storage Distributed Resource Scheduler (SDRS) がデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage DRS を無効にする必要があります。

複数のデータストアパスを指定することはできません。複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。

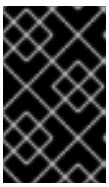
入力するクラスター名は、DNS レコードの設定時に指定したクラスター名と一致する必要があります。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

注記

3 ノードクラスターをインストールする場合は、必ず **compute.replicas** パラメーターを **0** に設定してください。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細については、「vSphere への 3 ノードクラスターのインストール」を参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- インストール設定パラメーター

23.2.4.4.1. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- architecture: amd64
  name:  <worker_node>
  platform: {}
  replicas: 3
controlPlane: (3)
  architecture: amd64
  name: <parent_node>
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: test (4)
platform:
  vsphere: (5)
    apiVIPs:
      - 10.0.0.1
    failureDomains: (6)
    - name: <failure_domain_name>
      region: <default_region_name>
```

```
      server: <fully_qualified_domain_name>
      topology:
        computeCluster: "/<datacenter>/host/<cluster>"
        datacenter: <datacenter>
        datastore: "/<datacenter>/datastore/<datastore>" 7
        networks:
        - <VM_Network_name>
        resourcePool: "/<datacenter>/host/<cluster>/Resources/<resourcePool>" 8
        folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>"
        tagIDs: 9
        - <tag_id> 10
      zone: <default_zone_name>
    ingressVIPs:
    - 10.0.0.2
    vcenters:
    - datacenters:
      - <datacenter>
      password: <password>
      port: 443
      server: <fully_qualified_domain_name>
      user: administrator@vsphere.local
    diskType: thin 11
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 3 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 DNS レコードに指定したクラスター名。

5 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

6 リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。

7 仮想マシンファイル、テンプレート、ISO イメージを保持する vSphere データストアへのパス。

重要

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion はデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage vMotion を無効にする必要があります。

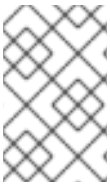
複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

8 オプション: マシン作成用の既存のリソースプールを提供します。値を指定しない場合、インストールプログラムは vSphere クラスターのルートリソースプールを使用します。

9 オプション: OpenShift Container Platform によって作成された各仮想マシンには、クラスターに固有の一意のタグが割り当てられます。割り当てられたタグにより、クラスターの使用停止時に、関連付けられた仮想マシンをインストールプログラムが識別して削除できるようになります。インストールプログラムによってプロビジョニングされた仮想マシンに割り当てる追加のタグ ID を最大 10 個までリストできます。

10 インストールプログラムによって関連付けられるタグの ID。たとえば、**urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL** です。タグ ID の決定の詳細は、vSphere Tags and Attributes ドキュメントを参照してください。

11 vSphere ディスクのプロビジョニング方法。



注記

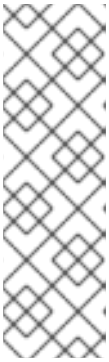
OpenShift Container Platform 4.12 以降では、**apiVIP** および **ingressVIP** 設定は非推奨です。代わりに、リスト形式を使用して、**apiVIPs** および **ingressVIPs** 設定に値を入力します。

23.2.4.4.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

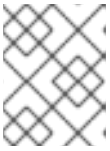
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

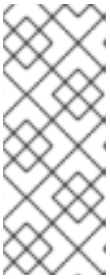
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

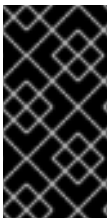
注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

23.2.4.4.3. VMware vCenter のリージョンとゾーンの設定

デフォルトのインストール設定ファイルを変更して、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできるようにします。

OpenShift Container Platform の以前のリリースのデフォルトの **install-config.yaml** ファイル設定は非推奨になりました。非推奨のデフォルト設定を引き続き使用できますが、**openshift-installer** により、設定ファイル内の非推奨のフィールドの使用を示す警告メッセージが表示されます。

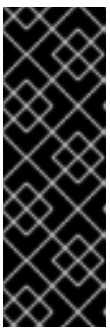


重要

この例では、**govc** コマンドを使用します。**govc** コマンドは、VMware から入手できるオープンソースコマンドです。Red Hat からは入手できません。Red Hat サポートチームは **govc** コマンドを保守していません。**govc** のダウンロードとインストールの手順については、VMware ドキュメント Web サイトを参照してください。

前提条件

- 既存の **install-config.yaml** インストール設定ファイルがあります。

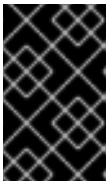


重要

VMware vCenter Server のデータセンターオブジェクトをプロビジョニングできるように、OpenShift Container Platform クラスターに少なくとも 1 つの障害ドメインを指定する必要があります。異なるデータセンター、クラスター、データストア、その他のコンポーネントに仮想マシンノードをプロビジョニングする必要がある場合は、複数の障害ドメインを指定することを検討してください。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

手順

1. 次の **govc** コマンドラインツールコマンドを入力して、**openshift-region** および **openshift-zone** vCenter タグカテゴリーを作成します。



重要

openshift-region および **openshift-zone** vCenter タグカテゴリーに異なる名前を指定すると、OpenShift Container Platform クラスターのインストールは失敗します。

```
$ govc tags.category.create -d "OpenShift region" openshift-region
```

```
$ govc tags.category.create -d "OpenShift zone" openshift-zone
```

2. クラスターをデプロイする各リージョン vSphere データセンターのリージョンタグを作成するには、ターミナルで次のコマンドを入力します。

```
$ govc tags.create -c <region_tag_category> <region_tag>
```

3. クラスターをデプロイする vSphere クラスターごとにゾーンタグを作成するには、次のコマンドを入力します。

```
$ govc tags.create -c <zone_tag_category> <zone_tag>
```

4. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにリージョンタグをアタッチします。

```
$ govc tags.attach -c <region_tag_category> <region_tag_1> /<datacenter_1>
```

5. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにゾーンタグをアタッチします。

```
$ govc tags.attach -c <zone_tag_category> <zone_tag_1> /<datacenter_1>/host/vcs-mdcnc-workload-1
```

6. インストールプログラムが含まれるディレクトリーに移動し、選択したインストール要件に従ってクラスターデプロイメントを初期化します。

vSphere センターで定義された複数のデータセンターを含むサンプル install-config.yaml ファイル

```
---
compute:
---
  vsphere:
     zones:
       - "<machine_pool_zone_1>"
       - "<machine_pool_zone_2>"
---
controlPlane:
---
```

```
vsphere:
      zones:
        - "<machine_pool_zone_1>"
        - "<machine_pool_zone_2>"
---
platform:
  vsphere:
    vcenters:
---
    datacenters:
      - <datacenter1_name>
      - <datacenter2_name>
    failureDomains:
    - name: <machine_pool_zone_1>
      region: <region_tag_1>
      zone: <zone_tag_1>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter1>
        computeCluster: "/<datacenter1>/host/<cluster1>"
        networks:
        - <VM_Network1_name>
        datastore: "/<datacenter1>/datastore/<datastore1>"
        resourcePool: "/<datacenter1>/host/<cluster1>/Resources/<resourcePool1>"
        folder: "/<datacenter1>/vm/<folder1>"
    - name: <machine_pool_zone_2>
      region: <region_tag_2>
      zone: <zone_tag_2>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter2>
        computeCluster: "/<datacenter2>/host/<cluster2>"
        networks:
        - <VM_Network2_name>
        datastore: "/<datacenter2>/datastore/<datastore2>"
        resourcePool: "/<datacenter2>/host/<cluster2>/Resources/<resourcePool2>"
        folder: "/<datacenter2>/vm/<folder2>"
---
```

23.2.4.5. ユーザー管理ロードバランサーのサービス

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。

重要

ユーザー管理ロードバランサーの設定は、ベンダーのロードバランサーによって異なります。

このセクションの情報と例は、ガイドラインのみを目的としています。ベンダーのロードバランサーに関する詳細は、ベンダーのドキュメントを参照してください。

Red Hat は、ユーザー管理ロードバランサーに対して次のサービスをサポートしています。

- Ingress Controller

- OpenShift API
- OpenShift MachineConfig API

ユーザー管理ロードバランサーに対して、これらのサービスの 1 つを設定するか、すべてを設定するかを選択できます。一般的な設定オプションは、Ingress Controller サービスのみを設定することです。次の図は、各サービスの詳細を示しています。

図23.1 OpenShift Container Platform 環境で動作する Ingress Controller を示すネットワークワークフローの例

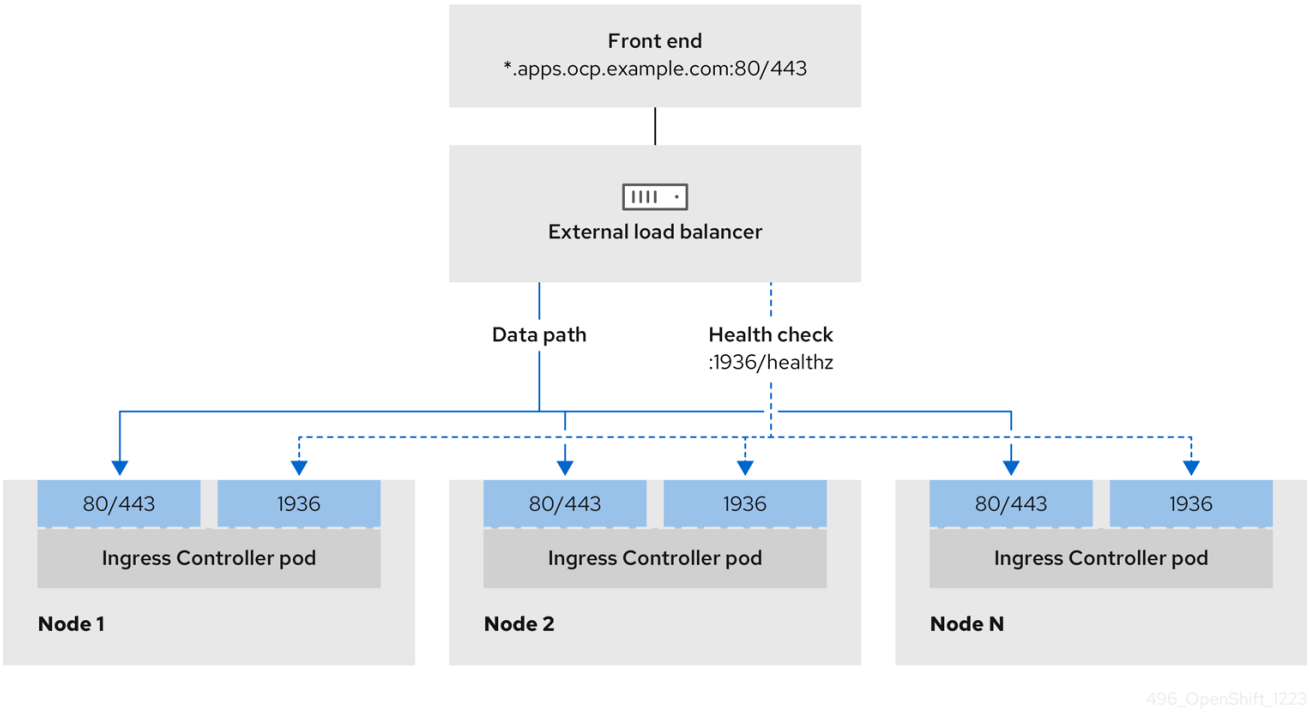


図23.2 OpenShift Container Platform 環境で動作する OpenShift API を示すネットワークワークフローの例

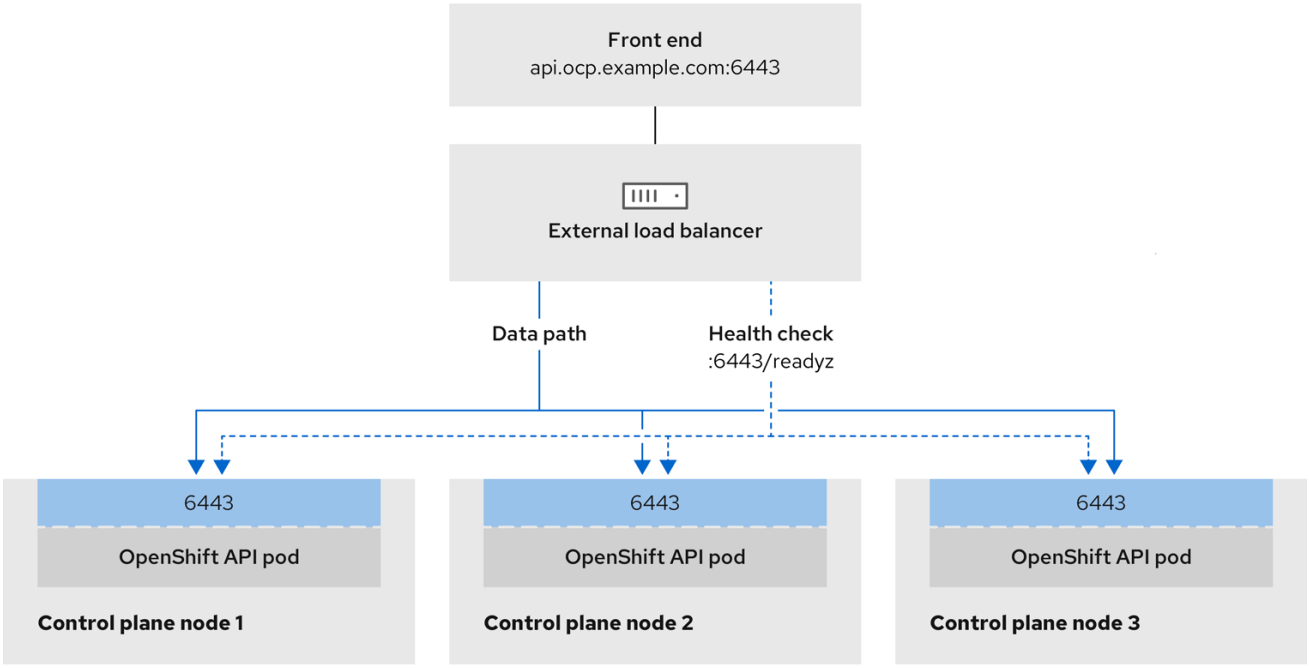
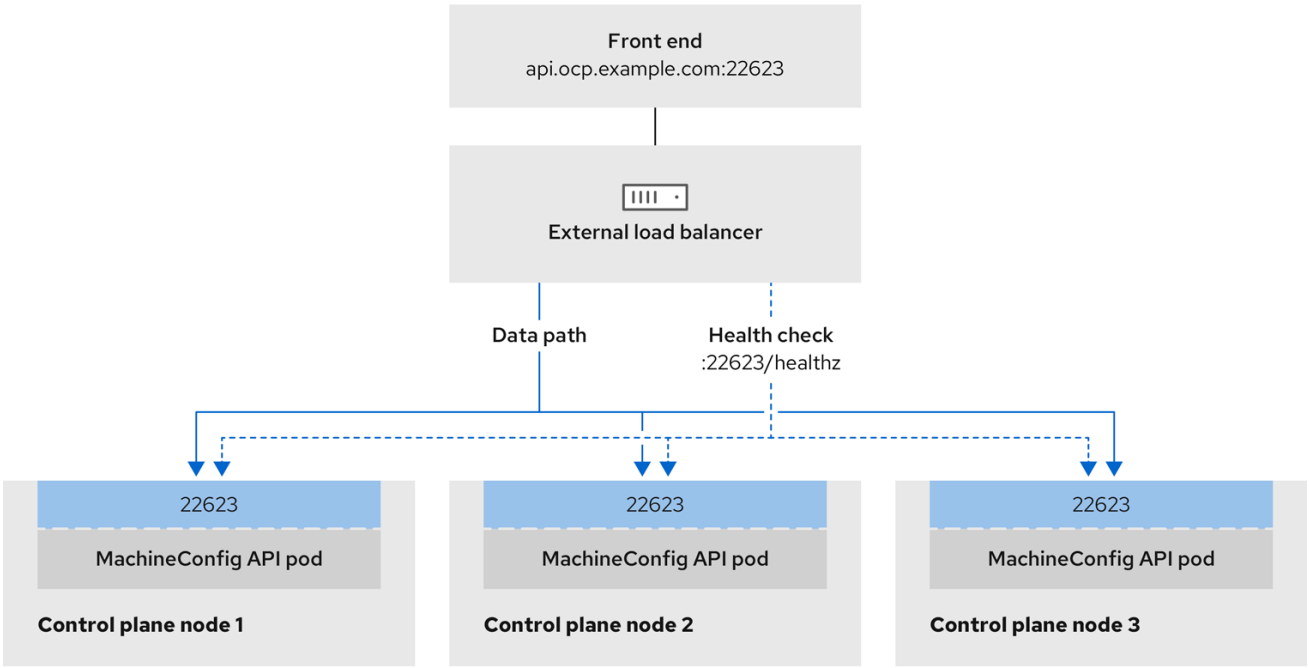


図23.3 OpenShift Container Platform 環境で動作する OpenShift MachineConfig API を示すネットワークワークフローの例



ユーザー管理ロードバランサーでは、次の設定オプションがサポートされています。

- ノードセレクターを使用して、Ingress Controller を特定のノードのセットにマッピングします。このセットの各ノードに静的 IP アドレスを割り当てるか、Dynamic Host Configuration Protocol (DHCP) から同じ IP アドレスを受け取るように各ノードを設定する必要があります。インフラストラクチャーノードは通常、このタイプの設定を受け取ります。

- サブネット上のすべての IP アドレスをターゲットにします。この設定では、ロードバランサーターゲットを再設定せずにネットワーク内でノードを作成および破棄できるため、メンテナンスオーバーヘッドを削減できます。**/27** や **/28** などの小規模なネットワーク上に設定されたマシンを使用して Ingress Pod をデプロイする場合、ロードバランサーのターゲットを簡素化できます。

  ヒント

  マシン config プールのリソースを確認することで、ネットワーク内に存在するすべての IP アドレスをリスト表示できます。

OpenShift Container Platform クラスターのユーザー管理ロードバランサーを設定する前に、以下の情報を考慮してください。

- フロントエンド IP アドレスの場合、フロントエンド IP アドレス、Ingress Controller のロードバランサー、および API ロードバランサーに同じ IP アドレスを使用できます。この機能については、ベンダーのドキュメントを確認してください。
- バックエンド IP アドレスの場合、ユーザー管理ロードバランサーの有効期間中に OpenShift Container Platform コントロールプレーンノードの IP アドレスが変更されないことを確認します。次のいずれかのアクションを実行すると、これを実現できます。
  - 各コントロールプレーンノードに静的 IP アドレスを割り当てます。
  - ノードが DHCP リースを要求するたびに、DHCP から同じ IP アドレスを受信するように各ノードを設定します。ベンダーによっては、DHCP リースは IP 予約または静的 DHCP 割り当ての形式になる場合があります。
- Ingress Controller バックエンドサービスのユーザー管理ロードバランサーで Ingress Controller を実行する各ノードを手動で定義します。たとえば、Ingress Controller が未定義のノードに移動すると、接続が停止する可能性があります。

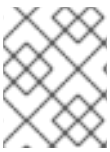
23.2.4.5.1. ユーザー管理ロードバランサーの設定

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。

重要

ユーザー管理ロードバランサーを設定する前に、ユーザー管理ロードバランサーのサービス" セクションを必ずお読みください。

ユーザー管理ロードバランサー用に設定するサービスに適用される次の前提条件をお読みください。



注記

クラスター上で実行される MetalLB は、ユーザー管理ロードバランサーとして機能します。

OpenShift API の前提条件

- フロントエンド IP アドレスを定義している。

- TCP ポート 6443 および 22623 は、ロードバランサーのフロントエンド IP アドレスで公開されている。以下の項目を確認します。
  - ポート 6443 が OpenShift API サービスにアクセスできる。
  - ポート 22623 が Ignition 起動設定をノードに提供できる。
- フロントエンド IP アドレスとポート 6443 へは、OpenShift Container Platform クラスターの外部の場所にいるシステムのすべてのユーザーがアクセスできる。
- フロントエンド IP アドレスとポート 22623 は、OpenShift Container Platform ノードからのみ到達できる。
- ロードバランサーバックエンドは、ポート 6443 および 22623 の OpenShift Container Platform コントロールプレーンノードと通信できる。

Ingress Controller の前提条件

- フロントエンド IP アドレスを定義している。
- TCP ポート 443 および 80 はロードバランサーのフロントエンド IP アドレスで公開されている。
- フロントエンドの IP アドレス、ポート 80、ポート 443 へは、OpenShift Container Platform クラスターの外部の場所にあるシステムの全ユーザーがアクセスできる。
- フロントエンドの IP アドレス、ポート 80、ポート 443 は、OpenShift Container Platform クラスターで動作するすべてのノードから到達できる。
- ロードバランサーバックエンドは、ポート 80、443、および 1936 で Ingress Controller を実行する OpenShift Container Platform ノードと通信できる。

ヘルスチェック URL 仕様の前提条件

ほとんどのロードバランサーは、サービスが使用可能か使用不可かを判断するヘルスチェック URL を指定して設定できまうｓ．OpenShift Container Platform は、OpenShift API、Machine Configuration API、および Ingress Controller バックエンドサービスのこれらのヘルスチェックを提供します。

次の例は、前にリスト表示したバックエンドサービスのヘルスチェック仕様を示しています。

Kubernetes API ヘルスチェック仕様の例

```
Path: HTTPS:6443/readyz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Machine Config API ヘルスチェック仕様の例

```
Path: HTTPS:22623/healthz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Ingress Controller のヘルスチェック仕様の例

```
Path: HTTP:1936/healthz/ready
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 5
Interval: 10
```

手順

1. HAProxy Ingress Controller を設定して、ポート 6443、22623、443、および 80 でロードバランサーからクラスターへのアクセスを有効化できるようにします。必要に応じて、HAProxy 設定で単一のサブネットの IP アドレスまたは複数のサブネットの IP アドレスを指定できます。

   #### 1つのサブネットをリストした HAProxy 設定の例

   ```
   # ...
   listen my-cluster-api-6443
       bind 192.168.1.100:6443
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /readyz
     http-check expect status 200
       server my-cluster-master-2 192.168.1.101:6443 check inter 10s rise 2 fall 2
       server my-cluster-master-0 192.168.1.102:6443 check inter 10s rise 2 fall 2
       server my-cluster-master-1 192.168.1.103:6443 check inter 10s rise 2 fall 2

   listen my-cluster-machine-config-api-22623
       bind 192.168.1.100:22623
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /healthz
     http-check expect status 200
       server my-cluster-master-2 192.168.1.101:22623 check inter 10s rise 2 fall 2
       server my-cluster-master-0 192.168.1.102:22623 check inter 10s rise 2 fall 2
       server my-cluster-master-1 192.168.1.103:22623 check inter 10s rise 2 fall 2

   listen my-cluster-apps-443
       bind 192.168.1.100:443
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /healthz/ready
     http-check expect status 200
       server my-cluster-worker-0 192.168.1.111:443 check port 1936 inter 10s rise 2 fall 2
       server my-cluster-worker-1 192.168.1.112:443 check port 1936 inter 10s rise 2 fall 2
       server my-cluster-worker-2 192.168.1.113:443 check port 1936 inter 10s rise 2 fall 2

   listen my-cluster-apps-80
     bind 192.168.1.100:80
   ```

```
  mode tcp
  balance roundrobin
 option httpchk
 http-check connect
 http-check send meth GET uri /healthz/ready
 http-check expect status 200
   server my-cluster-worker-0 192.168.1.111:80 check port 1936 inter 10s rise 2 fall 2
   server my-cluster-worker-1 192.168.1.112:80 check port 1936 inter 10s rise 2 fall 2
   server my-cluster-worker-2 192.168.1.113:80 check port 1936 inter 10s rise 2 fall 2
# ...
```

複数のサブネットをリストした HAProxy 設定の例

```
# ...
listen api-server-6443
   bind *:6443
   mode tcp
     server master-00 192.168.83.89:6443 check inter 1s
     server master-01 192.168.84.90:6443 check inter 1s
     server master-02 192.168.85.99:6443 check inter 1s
     server bootstrap 192.168.80.89:6443 check inter 1s

listen machine-config-server-22623
   bind *:22623
   mode tcp
     server master-00 192.168.83.89:22623 check inter 1s
     server master-01 192.168.84.90:22623 check inter 1s
     server master-02 192.168.85.99:22623 check inter 1s
     server bootstrap 192.168.80.89:22623 check inter 1s

listen ingress-router-80
   bind *:80
   mode tcp
   balance source
     server worker-00 192.168.83.100:80 check inter 1s
     server worker-01 192.168.83.101:80 check inter 1s

listen ingress-router-443
   bind *:443
   mode tcp
   balance source
     server worker-00 192.168.83.100:443 check inter 1s
     server worker-01 192.168.83.101:443 check inter 1s

listen ironic-api-6385
   bind *:6385
   mode tcp
   balance source
     server master-00 192.168.83.89:6385 check inter 1s
     server master-01 192.168.84.90:6385 check inter 1s
     server master-02 192.168.85.99:6385 check inter 1s
     server bootstrap 192.168.80.89:6385 check inter 1s

listen inspector-api-5050
   bind *:5050
   mode tcp
```

```
    balance source
      server master-00 192.168.83.89:5050 check inter 1s
      server master-01 192.168.84.90:5050 check inter 1s
      server master-02 192.168.85.99:5050 check inter 1s
      server bootstrap 192.168.80.89:5050 check inter 1s
# ...
```

2. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーとそのリソースが動作していることを確認します。

   a. 次のコマンドを実行して応答を観察し、クラスターマシン設定 API が Kubernetes API サーバーリソースにアクセスできることを確認します。

      ```
      $ curl https://<loadbalancer_ip_address>:6443/version --insecure
      ```

      設定が正しい場合は、応答として JSON オブジェクトを受信します。

      ```
      {
        "major": "1",
        "minor": "11+",
        "gitVersion": "v1.11.0+ad103ed",
        "gitCommit": "ad103ed",
        "gitTreeState": "clean",
        "buildDate": "2019-01-09T06:44:10Z",
        "goVersion": "go1.10.3",
        "compiler": "gc",
        "platform": "linux/amd64"
      }
      ```

   b. 次のコマンドを実行して出力を確認し、クラスターマシン設定 API がマシン設定サーバーリソースからアクセスできることを確認します。

      ```
      $ curl -v https://<loadbalancer_ip_address>:22623/healthz --insecure
      ```

      設定が正しい場合、コマンドの出力には次の応答が表示されます。

      ```
      HTTP/1.1 200 OK
      Content-Length: 0
      ```

   c. 次のコマンドを実行して出力を確認し、コントローラーがポート 80 の Ingress Controller リソースにアクセスできることを確認します。

      ```
      $ curl -I -L -H "Host: console-openshift-console.apps.<cluster_name>.<base_domain>"
      http://<load_balancer_front_end_IP_address>
      ```

      設定が正しい場合、コマンドの出力には次の応答が表示されます。

      ```
      HTTP/1.1 302 Found
      content-length: 0
      location: https://console-openshift-console.apps.ocp4.private.opequon.net/
      cache-control: no-cache
      ```

   d. 次のコマンドを実行して出力を確認し、コントローラーがポート 443 の Ingress Controller リソースにアクセスできることを確認します。

```
$ curl -I -L --insecure --resolve console-openshift-console.apps.<cluster_name>.
<base_domain>:443:<Load Balancer Front End IP Address> https://console-openshift-
console.apps.<cluster_name>.<base_domain>
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

3. ユーザー管理ロードバランサーのフロントエンド IP アドレスをターゲットにするようにクラスターの DNS レコードを設定します。ロードバランサー経由で、クラスター API およびアプリケーションの DNS サーバーのレコードを更新する必要があります。

変更された DNS レコードの例

```
<load_balancer_ip_address>  A  api.<cluster_name>.<base_domain>
A record pointing to Load Balancer Front End
```

```
<load_balancer_ip_address>   A apps.<cluster_name>.<base_domain>
A record pointing to Load Balancer Front End
```

重要

DNS の伝播では、各 DNS レコードが使用可能になるまでに時間がかかる場合があります。各レコードを検証する前に、各 DNS レコードが伝播されることを確認してください。

4. OpenShift Container Platform クラスターでユーザー管理ロードバランサーを使用するには、クラスターの **install-config.yaml** ファイルで次の設定を指定する必要があります。

```
# ...
platform:
  vsphere:
    loadBalancer:
      type: UserManaged 1
      apiVIPs:
      - <api_ip> 2
```

```
    ingressVIPs:
    - <ingress_ip> 3
# ...
```

1 クラスターのユーザー管理ロードバランサーを指定するには、**type** パラメーターに **UserManaged** を設定します。パラメーターのデフォルトは **OpenShiftManagedDefault** で、これはデフォルトの内部ロードバランサーを示します。**openshift-kni-infra** namespace で定義されたサービスの場合、ユーザー管理ロードバランサーは **coredns** サービスをクラスター内の Pod にデプロイできますが、**keepalived** および **haproxy** サービスは無視します。

2 ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。Kubernetes API がユーザー管理ロードバランサーと通信できるように、ユーザー管理ロードバランサーのパブリック IP アドレスを指定します。

3 ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。ユーザー管理ロードバランサーのパブリック IP アドレスを指定して、ユーザー管理ロードバランサーがクラスターの Ingress トラフィックを管理できるようにします。

検証

1. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーと DNS レコード設定が動作していることを確認します。

   a. 次のコマンドを実行して出力を確認し、クラスター API にアクセスできることを確認します。

   ```
   $ curl https://api.<cluster_name>.<base_domain>:6443/version --insecure
   ```

   設定が正しい場合は、応答として JSON オブジェクトを受信します。

   ```
   {
     "major": "1",
     "minor": "11+",
     "gitVersion": "v1.11.0+ad103ed",
     "gitCommit": "ad103ed",
     "gitTreeState": "clean",
     "buildDate": "2019-01-09T06:44:10Z",
     "goVersion": "go1.10.3",
     "compiler": "gc",
     "platform": "linux/amd64"
     }
   ```

   b. 次のコマンドを実行して出力を確認し、クラスターマシン設定にアクセスできることを確認します。

   ```
   $ curl -v https://api.<cluster_name>.<base_domain>:22623/healthz --insecure
   ```

   設定が正しい場合、コマンドの出力には次の応答が表示されます。

   ```
   HTTP/1.1 200 OK
   Content-Length: 0
   ```

c. 以下のコマンドを実行して出力を確認し、ポートで各クラスターアプリケーションにアクセスできることを確認します。

```
$ curl http://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.<cluster-name>.<base domain>/
cache-control: no-cacheHTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=39HoZgztDnzjJkq/JuLJMeoKNXlfiVv2YgZc09c3TBOBU4Nl6kDXaJH1LdicNhN1UsQ
Wzon4Dor9GWGfopaTEQ==; Path=/; Secure
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Tue, 17 Nov 2020 08:42:10 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=9b714eb87e93cf34853e87a92d6894be; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

d. 次のコマンドを実行して出力を確認し、ポート 443 で各クラスターアプリケーションにアクセスできることを確認します。

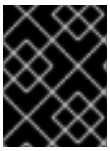
```
$ curl https://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

23.2.4.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。
- オプション: クラスターを作成する前に、デフォルトのロードバランサーの代わりに外部ロードバランサーを設定します。

重要

インストールプログラムに API および Ingress 静的アドレスを指定する必要はありません。この設定を選択した場合は、追加のアクションを実行して、参照される各 vSphere サブネットから IP アドレスを受け入れるネットワークターゲットを定義する必要があります。「ユーザー管理ロードバランサーの設定」セクションを参照してください。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

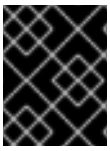
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

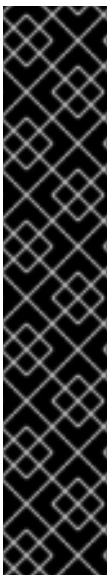


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

23.2.4.7. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

23.2.4.8. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

23.2.4.8.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

23.2.4.8.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

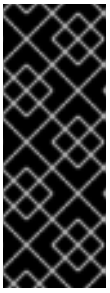
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

23.2.4.8.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Data Foundation など、クラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- "100Gi" の容量が必要です。

重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリックストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

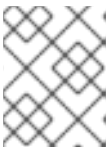
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

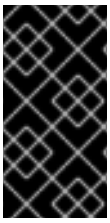
```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

23.2.4.8.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、**1** つのレプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

3 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

4 永続ボリューム要求 (PVC) のサイズ。

b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

正しい PVC を参照するようにレジストリーストレージを設定する手順は、vSphere のレジストリーの設定 を参照してください。

23.2.4.9. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

23.2.4.10. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

23.2.5. ネットワークのカスタマイズによる vSphere へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、installer-provisioned infrastructure とカスタマイズした設定オプションを使用して、VMware vSphere インスタンスにクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

23.2.5.1. 前提条件

- installer-provisioned infrastructure を使用したクラスターのインストールの準備 のタスクを完了した。
- VMware プラットフォームのライセンスを確認した。Red Hat は VMware ライセンスに制限を設けていませんが、一部の VMware インフラストラクチャーコンポーネントにはライセンスが必要です。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターの 永続ストレージ をプロビジョニングした。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform インストーラーは、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

23.2.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

23.2.5.3. VMware vSphere のリージョンとゾーンの有効化

OpenShift Container Platform クラスターを、単一の VMware vCenter で実行される複数の vSphere データセンターにデプロイできます。各データセンターは複数のクラスターを実行できます。この設定により、クラスターの障害を引き起こす可能性のあるハードウェア障害やネットワーク停止のリスクが軽減されます。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

重要

VMware vSphere のリージョンおよびゾーンの有効化機能には、クラスター内のデフォルトのストレージドライバーとして vSphere Container Storage Interface (CSI) ドライバーが必要です。そのため、この機能は新しくインストールされたクラスターでのみ使用できます。

以前のリリースからアップグレードされたクラスターの場合は、クラスターの CSI 自動移行を有効にする必要があります。その後、アップグレードされたクラスターに対して複数のリージョンとゾーンを設定できます。

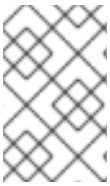
デフォルトのインストール設定では、クラスターが単一の vSphere データセンターにデプロイされます。クラスターを複数の vSphere データセンターにデプロイする場合は、リージョンおよびゾーン機能を有効にするインストール設定ファイルを作成する必要があります。

デフォルトの **install-config.yaml** ファイルには **vcenters フィールド** と **FailureDomains** フィールドが含まれており、OpenShift Container Platform クラスターに複数の vSphere データセンターとクラスターを指定できます。単一のデータセンターで設定される vSphere 環境に OpenShift Container Platform クラスターをインストールする場合は、これらのフィールドを空白のままにすることができます。

次のリストでは、クラスターのゾーンとリージョンの定義に関連する用語について説明します。

- 障害ドメイン: リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。
- リージョン: vCenter データセンターを指定します。リージョンを定義するには、**openshift-region** タグカテゴリーのタグを使用します。

- ゾーン: vCenter クラスターを指定します。ゾーンを定義するには、**openshift-zone** タグカテゴリーのタグを使用します。



注記

install-config.yaml ファイルで複数の障害ドメインを指定する予定がある場合は、設定ファイルを作成する前に、タグカテゴリー、ゾーンタグ、およびリージョンタグを作成する必要があります。

リージョンを表す vCenter データセンターごとに vCenter タグを作成する必要があります。さらに、データセンターで実行されるクラスターごとに、ゾーンを表す vCenter タグを作成する必要があります。タグを作成した後、各タグをそれぞれのデータセンターとクラスターにアタッチする必要があります。

次の表は、単一の VMware vCenter で実行されている複数の vSphere データセンターを含む設定のリージョン、ゾーン、タグ間の関係の例を示しています。

データセンター (リージョン)	クラスター (ゾーン)	タグ
米国東部	us-east-1	us-east-1a
		us-east-1b
	us-east-2	us-east-2a
		us-east-2b
us-west	us-west-1	us-west-1a
		us-west-1b
	us-west-2	us-west-2a
		us-west-2b

関連情報

- 追加の VMware vSphere 設定パラメーター
- 非推奨の VMware vSphere 設定パラメーター
- vSphere の自動移行
- VMware vSphere CSI ドライバー Operator

23.2.5.4. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

      iii. vCenter インスタンスの名前を指定します。

      iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
      インストールプログラムは vCenter インスタンスに接続します。

      v. 接続する vCenter インスタンスのデータセンターを選択します。

注記

インストール設定ファイルを作成した後、そのファイルを変更して複数の vSphere データセンター環境を作成できます。これは、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできることを意味します。この環境の作成の詳細については、**VMware vSphere のリージョンとゾーンの有効化** を参照してください。

vi. 使用するデフォルトの vCenter データストアを選択します。

警告

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion を使用する Storage Distributed Resource Scheduler (SDRS) がデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage DRS を無効にする必要があります。

複数のデータストアパスを指定することはできません。複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

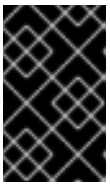
x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。
入力するクラスター名は、DNS レコードの設定時に指定したクラスター名と一致する必要があります。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- インストール設定パラメーター

23.2.5.4.1. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- architecture: amd64
  name:  <worker_node>
  platform: {}
  replicas: 3
controlPlane: (3)
  architecture: amd64
  name: <parent_node>
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: test (4)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes (5)
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere: (6)
    apiVIPs:
      - 10.0.0.1
    failureDomains: (7)
    - name: <failure_domain_name>
      region: <default_region_name>
      server: <fully_qualified_domain_name>
      topology:
        computeCluster: "/<datacenter>/host/<cluster>"
        datacenter: <datacenter>
        datastore: "/<datacenter>/datastore/<datastore>" (8)
        networks:
        - <VM_Network_name>
```

```
      resourcePool: "/<datacenter>/host/<cluster>/Resources/<resourcePool>" 9
      folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>"
      tagIDs: 10
      - <tag_id> 11
    zone: <default_zone_name>
  ingressVIPs:
  - 10.0.0.2
  vcenters:
  - datacenters:
    - <datacenter>
    password: <password>
    port: 443
    server: <fully_qualified_domain_name>
    user: administrator@vsphere.local
  diskType: thin 12
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 3 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 DNS レコードに指定したクラスター名。

6 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

7 リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。

8 仮想マシンファイル、テンプレート、ISO イメージを保持する vSphere データストアへのパス。

重要

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion はデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage vMotion を無効にする必要があります。

複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

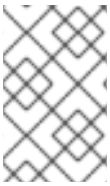
9 オプション: マシン作成用の既存のリソースプールを提供します。値を指定しない場合、インストールプログラムは vSphere クラスターのルートリソースプールを使用します。

10 オプション: OpenShift Container Platform によって作成された各仮想マシンには、クラスターに

11 インストールプログラムによって関連付けられるタグの ID。たとえば、**urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL** で

12 vSphere ディスクのプロビジョニング方法。

5 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。



注記

OpenShift Container Platform 4.12 以降では、**apiVIP** および **ingressVIP** 設定は非推奨です。代わりに、リスト形式を使用して、**apiVIPs** および **ingressVIPs** 設定に値を入力します。

23.2.5.4.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
```

```
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

23.2.5.4.3. オプション: デュアルスタックネットワークを使用したデプロイ

OpenShift Container Platform クラスターのデュアルスタックネットワークでは、クラスターノードの IPv4 および IPv6 アドレスエンドポイントを設定できます。クラスターノードの IPv4 および IPv6 アドレスエンドポイントを設定するには、**install-config.yaml** ファイルで **machineNetwork**、**clusterNetwork**、および **serviceNetwork** 設定を編集します。それぞれの設定には、それぞれ 2 つの CIDR エントリーが必要です。プライマリーアドレスファミリーとして IPv4 ファミリーを持つクラスターの場合は、最初に IPv4 設定を指定します。プライマリーアドレスファミリーとして IPv6 ファミリーを持つクラスターの場合は、最初に IPv6 設定を指定します。

```
machineNetwork:
- cidr: {{ extcidrnet }}
- cidr: {{ extcidrnet6 }}
clusterNetwork:
- cidr: 10.128.0.0/14
  hostPrefix: 23
- cidr: fd02::/48
  hostPrefix: 64
serviceNetwork:
- 172.30.0.0/16
- fd03::/112
```

IPv4 および IPv6 アドレスを使用するアプリケーションのクラスターへのインターフェイスを提供するには、Ingress VIP および API VIP サービスの IPv4 および IPv6 仮想 IP (VIP) アドレスエンドポイントを設定します。IPv4 および IPv6 アドレスエンドポイントを設定するには、**install-config.yaml** ファイルで **apiVIPs** および **ingressVIPs** 設定を編集します。**apiVIPs** および **ingressVIPs** 設定では、リスト形式を使用します。リストの順序は、各サービスのプライマリーおよびセカンダリー VIP アドレスを示しています。

```
platform:
  vsphere:
    apiVIPs:
      - <api_ipv4>
      - <api_ipv6>
    ingressVIPs:
      - <wildcard_ipv4>
      - <wildcard_ipv6>
```

注記

デュアルスタックネットワーク設定のクラスターの場合、IPv4 アドレスと IPv6 アドレスの両方を同じインターフェイスに割り当てる必要があります。

23.2.5.4.4. VMware vCenter のリージョンとゾーンの設定

デフォルトのインストール設定ファイルを変更して、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできるようにします。

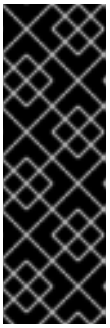
OpenShift Container Platform の以前のリリースのデフォルトの **install-config.yaml** ファイル設定は非推奨になりました。非推奨のデフォルト設定を引き続き使用できますが、**openshift-installer** により、設定ファイル内の非推奨のフィールドの使用を示す警告メッセージが表示されます。

重要

この例では、**govc** コマンドを使用します。**govc** コマンドは、VMware から入手できるオープンソースコマンドです。Red Hat からは入手できません。Red Hat サポートチームは **govc** コマンドを保守していません。**govc** のダウンロードとインストールの手順については、VMware ドキュメント Web サイトを参照してください。

前提条件

- 既存の **install-config.yaml** インストール設定ファイルがあります。



重要

VMware vCenter Server のデータセンターオブジェクトをプロビジョニングできるように、OpenShift Container Platform クラスターに少なくとも 1 つの障害ドメインを指定する必要があります。異なるデータセンター、クラスター、データストア、その他のコンポーネントに仮想マシンノードをプロビジョニングする必要がある場合は、複数の障害ドメインを指定することを検討してください。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

手順

1. 次の **govc** コマンドラインツールコマンドを入力して、**openshift-region** および **openshift-zone** vCenter タグカテゴリーを作成します。

重要

openshift-region および **openshift-zone** vCenter タグカテゴリーに異なる名前を指定すると、OpenShift Container Platform クラスターのインストールは失敗します。

```
$ govc tags.category.create -d "OpenShift region" openshift-region
```

```
$ govc tags.category.create -d "OpenShift zone" openshift-zone
```

2. クラスターをデプロイする各リージョン vSphere データセンターのリージョンタグを作成するには、ターミナルで次のコマンドを入力します。

```
$ govc tags.create -c <region_tag_category> <region_tag>
```

3. クラスターをデプロイする vSphere クラスターごとにゾーンタグを作成するには、次のコマンドを入力します。

```
$ govc tags.create -c <zone_tag_category> <zone_tag>
```

4. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにリージョンタグをアタッチします。

```
$ govc tags.attach -c <region_tag_category> <region_tag_1> /<datacenter_1>
```

5. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにゾーンタグをアタッチします。

```
$ govc tags.attach -c <zone_tag_category> <zone_tag_1> /<datacenter_1>/host/vcs-mdcnc-workload-1
```

6. インストールプログラムが含まれるディレクトリーに移動し、選択したインストール要件に従ってクラスターデプロイメントを初期化します。

vSphere センターで定義された複数のデータセンターを含むサンプル install-config.yaml ファイル

```
---
compute:
---
  vsphere:
      zones:
        - "<machine_pool_zone_1>"
        - "<machine_pool_zone_2>"
---
controlPlane:
---
vsphere:
      zones:
        - "<machine_pool_zone_1>"
        - "<machine_pool_zone_2>"
---
platform:
  vsphere:
    vcenters:
---
    datacenters:
      - <datacenter1_name>
      - <datacenter2_name>
    failureDomains:
    - name: <machine_pool_zone_1>
      region: <region_tag_1>
      zone: <zone_tag_1>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter1>
        computeCluster: "/<datacenter1>/host/<cluster1>"
        networks:
        - <VM_Network1_name>
        datastore: "/<datacenter1>/datastore/<datastore1>"
        resourcePool: "/<datacenter1>/host/<cluster1>/Resources/<resourcePool1>"
        folder: "/<datacenter1>/vm/<folder1>"
    - name: <machine_pool_zone_2>
      region: <region_tag_2>
      zone: <zone_tag_2>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter2>
        computeCluster: "/<datacenter2>/host/<cluster2>"
        networks:
```

```
        - <VM_Network2_name>
        datastore: "/<datacenter2>/datastore/<datastore2>"
        resourcePool: "/<datacenter2>/host/<cluster2>/Resources/<resourcePool2>"
        folder: "/<datacenter2>/vm/<folder2>"
---
```

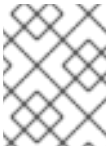
23.2.5.5. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

重要

CIDR 範囲 **172.17.0.0/16** は libVirt によって予約されています。この範囲、またはこの範囲と重複する範囲をクラスター内のネットワークに使用することはできません。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 でネットワークプラグインをさらにカスタマイズできます。

23.2.5.6. 高度なネットワーク設定の指定

ネットワークプラグインに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 次の例のように、**cluster-network-03-config.yml** ファイルでクラスターの高度なネットワーク設定を指定します。

   OVN-Kubernetes ネットワークプロバイダーを有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

5. コントロールプレーンマシンおよび compute machineSets を定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- MachineSet ファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

23.2.5.7. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

23.2.5.7.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表23.7 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 `spec:` `  clusterNetwork:` `  - cidr: 10.128.0.0/19` `  hostPrefix: 23` `  - cidr: 10.128.32.0/19` `  hostPrefix: 23`

フィールド	型	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表23.8 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表23.9 **ovnKubernetesConfig** オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。  **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表23.10 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表23.11 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表23.12 **policyAuditConfig オブジェクト**

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表23.13 **gatewayConfig** オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

フィールド	型	説明
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表23.14 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表23.15 **gatewayConfig.ipv6** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

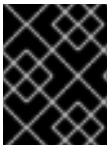
表23.16 **ipsecConfig** オブジェクト

フィールド	型	説明

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```

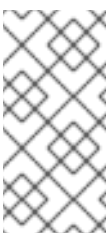


重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表23.17 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	型	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

23.2.5.8. ユーザー管理ロードバランサーのサービス

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。

重要

ユーザー管理ロードバランサーの設定は、ベンダーのロードバランサーによって異なります。

このセクションの情報と例は、ガイドラインのみを目的としています。ベンダーのロードバランサーに関する詳細は、ベンダーのドキュメントを参照してください。

Red Hat は、ユーザー管理ロードバランサーに対して次のサービスをサポートしています。

- Ingress Controller
- OpenShift API
- OpenShift MachineConfig API

ユーザー管理ロードバランサーに対して、これらのサービスの 1 つを設定するか、すべてを設定するかを選択できます。一般的な設定オプションは、Ingress Controller サービスのみを設定することです。次の図は、各サービスの詳細を示しています。

図23.4 OpenShift Container Platform 環境で動作する Ingress Controller を示すネットワークワークフローの例

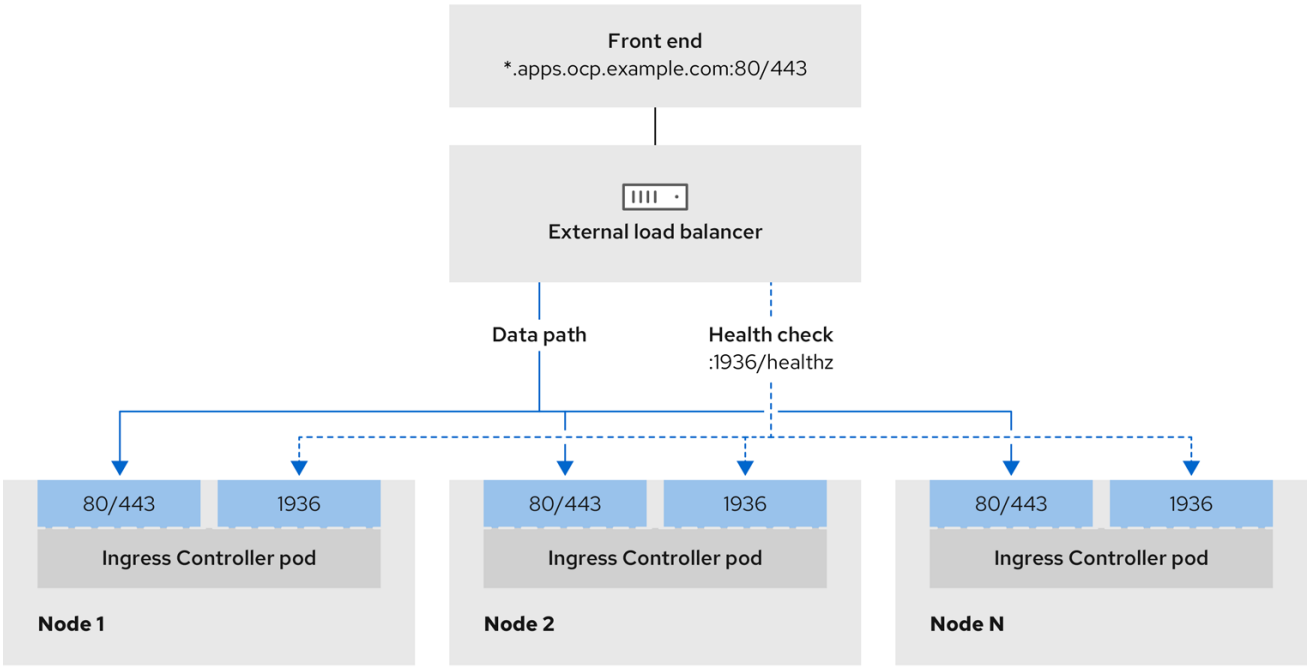


図23.5 OpenShift Container Platform 環境で動作する OpenShift API を示すネットワークワークフローの例

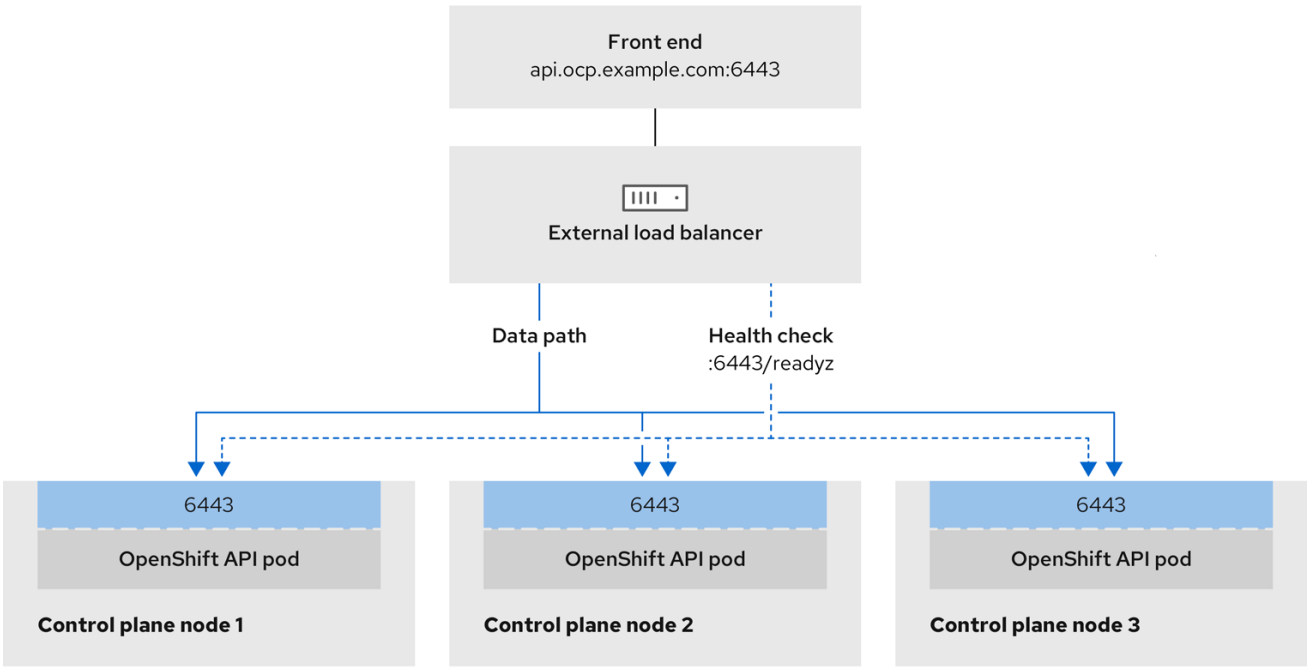
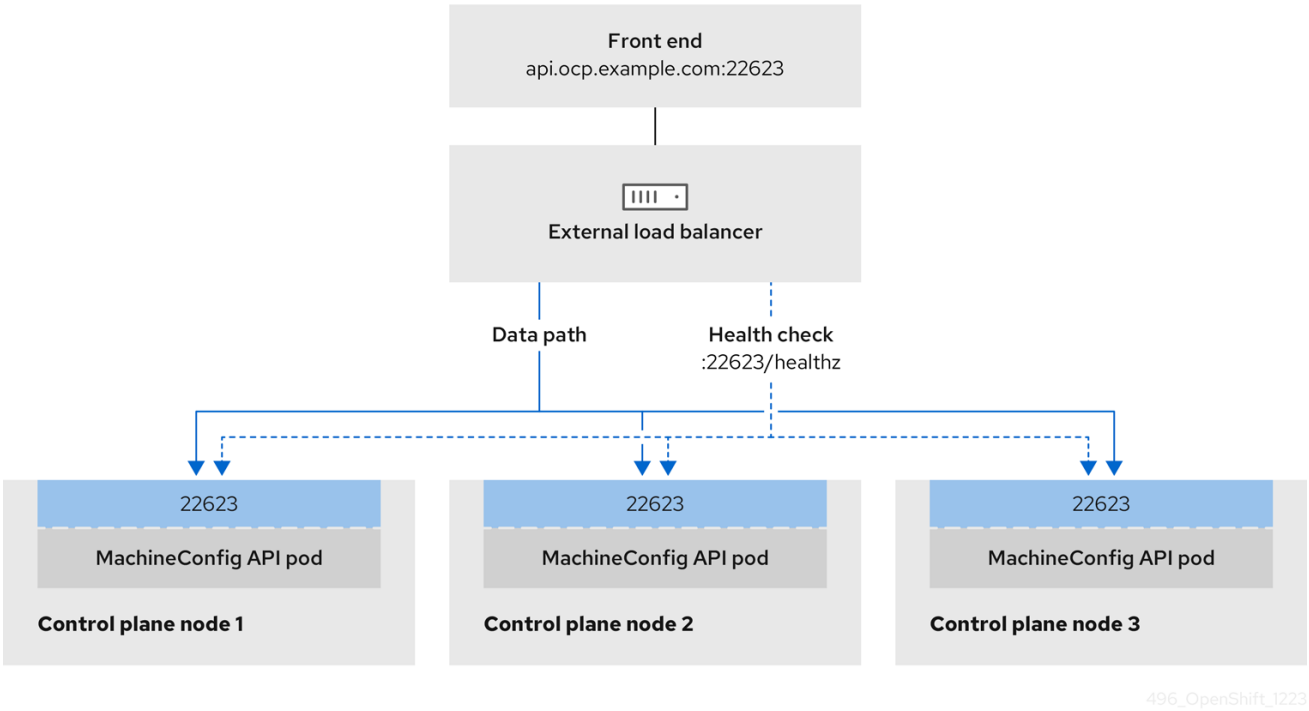


図23.6 OpenShift Container Platform 環境で動作する OpenShift MachineConfig API を示すネットワークワークフローの例



ユーザー管理ロードバランサーでは、次の設定オプションがサポートされています。

- ノードセレクターを使用して、Ingress Controller を特定のノードのセットにマッピングします。このセットの各ノードに静的 IP アドレスを割り当てるか、Dynamic Host Configuration Protocol (DHCP) から同じ IP アドレスを受け取るように各ノードを設定する必要があります。インフラストラクチャーノードは通常、このタイプの設定を受け取ります。
- サブネット上のすべての IP アドレスをターゲットにします。この設定では、ロードバランサーターゲットを再設定せずにネットワーク内でノードを作成および破棄できるため、メンテナンスオーバーヘッドを削減できます。**/27** や **/28** などの小規模なネットワーク上に設定されたマシンを使用して Ingress Pod をデプロイする場合、ロードバランサーのターゲットを簡素化できます。

  ヒント

  マシン config プールのリソースを確認することで、ネットワーク内に存在するすべての IP アドレスをリスト表示できます。

OpenShift Container Platform クラスターのユーザー管理ロードバランサーを設定する前に、以下の情報を考慮してください。

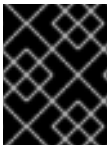
- フロントエンド IP アドレスの場合、フロントエンド IP アドレス、Ingress Controller のロードバランサー、および API ロードバランサーに同じ IP アドレスを使用できます。この機能については、ベンダーのドキュメントを確認してください。
- バックエンド IP アドレスの場合、ユーザー管理ロードバランサーの有効期間中に OpenShift Container Platform コントロールプレーンノードの IP アドレスが変更されないことを確認します。次のいずれかのアクションを実行すると、これを実現できます。
  - 各コントロールプレーンノードに静的 IP アドレスを割り当てます。

- ノードが DHCP リースを要求するたびに、DHCP から同じ IP アドレスを受信するように各ノードを設定します。ベンダーによっては、DHCP リースは IP 予約または静的 DHCP 割り当ての形式になる場合があります。

- Ingress Controller バックエンドサービスのユーザー管理ロードバランサーで Ingress Controller を実行する各ノードを手動で定義します。たとえば、Ingress Controller が未定義のノードに移動すると、接続が停止する可能性があります。

23.2.5.8.1. ユーザー管理ロードバランサーの設定

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。



重要

ユーザー管理ロードバランサーを設定する前に、ユーザー管理ロードバランサーのサービス" セクションを必ずお読みください。

ユーザー管理ロードバランサー用に設定するサービスに適用される次の前提条件をお読みください。

注記

クラスター上で実行される MetalLB は、ユーザー管理ロードバランサーとして機能します。

OpenShift API の前提条件

- フロントエンド IP アドレスを定義している。

- TCP ポート 6443 および 22623 は、ロードバランサーのフロントエンド IP アドレスで公開されている。以下の項目を確認します。

  - ポート 6443 が OpenShift API サービスにアクセスできる。

  - ポート 22623 が Ignition 起動設定をノードに提供できる。

- フロントエンド IP アドレスとポート 6443 へは、OpenShift Container Platform クラスターの外部の場所にいるシステムのすべてのユーザーがアクセスできる。

- フロントエンド IP アドレスとポート 22623 は、OpenShift Container Platform ノードからのみ到達できる。

- ロードバランサーバックエンドは、ポート 6443 および 22623 の OpenShift Container Platform コントロールプレーンノードと通信できる。

Ingress Controller の前提条件

- フロントエンド IP アドレスを定義している。

- TCP ポート 443 および 80 はロードバランサーのフロントエンド IP アドレスで公開されている。

- フロントエンドの IP アドレス、ポート 80、ポート 443 へは、OpenShift Container Platform クラスターの外部の場所にあるシステムの全ユーザーがアクセスできる。

- フロントエンドの IP アドレス、ポート 80、ポート 443 は、OpenShift Container Platform クラスターで動作するすべてのノードから到達できる。
- ロードバランサーバックエンドは、ポート 80、443、および 1936 で Ingress Controller を実行する OpenShift Container Platform ノードと通信できる。

ヘルスチェック URL 仕様の前提条件

ほとんどのロードバランサーは、サービスが使用可能か使用不可かを判断するヘルスチェック URL を指定して設定できまうｓ．OpenShift Container Platform は、OpenShift API、Machine Configuration API、および Ingress Controller バックエンドサービスのこれらのヘルスチェックを提供します。

次の例は、前にリスト表示したバックエンドサービスのヘルスチェック仕様を示しています。

Kubernetes API ヘルスチェック仕様の例

```
Path: HTTPS:6443/readyz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Machine Config API ヘルスチェック仕様の例

```
Path: HTTPS:22623/healthz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Ingress Controller のヘルスチェック仕様の例

```
Path: HTTP:1936/healthz/ready
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 5
Interval: 10
```

手順

1. HAProxy Ingress Controller を設定して、ポート 6443、22623、443、および 80 でロードバランサーからクラスターへのアクセスを有効化できるようにします。必要に応じて、HAProxy 設定で単一のサブネットの IP アドレスまたは複数のサブネットの IP アドレスを指定できます。

   #### 1つのサブネットをリストした HAProxy 設定の例

   ```
   # ...
   listen my-cluster-api-6443
       bind 192.168.1.100:6443
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /readyz
   ```

```
  http-check expect status 200
    server my-cluster-master-2 192.168.1.101:6443 check inter 10s rise 2 fall 2
    server my-cluster-master-0 192.168.1.102:6443 check inter 10s rise 2 fall 2
    server my-cluster-master-1 192.168.1.103:6443 check inter 10s rise 2 fall 2

listen my-cluster-machine-config-api-22623
    bind 192.168.1.100:22623
    mode tcp
    balance roundrobin
  option httpchk
  http-check connect
  http-check send meth GET uri /healthz
  http-check expect status 200
    server my-cluster-master-2 192.168.1.101:22623 check inter 10s rise 2 fall 2
    server my-cluster-master-0 192.168.1.102:22623 check inter 10s rise 2 fall 2
    server my-cluster-master-1 192.168.1.103:22623 check inter 10s rise 2 fall 2

listen my-cluster-apps-443
    bind 192.168.1.100:443
    mode tcp
    balance roundrobin
  option httpchk
  http-check connect
  http-check send meth GET uri /healthz/ready
  http-check expect status 200
    server my-cluster-worker-0 192.168.1.111:443 check port 1936 inter 10s rise 2 fall 2
    server my-cluster-worker-1 192.168.1.112:443 check port 1936 inter 10s rise 2 fall 2
    server my-cluster-worker-2 192.168.1.113:443 check port 1936 inter 10s rise 2 fall 2

listen my-cluster-apps-80
   bind 192.168.1.100:80
   mode tcp
   balance roundrobin
  option httpchk
  http-check connect
  http-check send meth GET uri /healthz/ready
  http-check expect status 200
    server my-cluster-worker-0 192.168.1.111:80 check port 1936 inter 10s rise 2 fall 2
    server my-cluster-worker-1 192.168.1.112:80 check port 1936 inter 10s rise 2 fall 2
    server my-cluster-worker-2 192.168.1.113:80 check port 1936 inter 10s rise 2 fall 2
# ...
```

複数のサブネットをリストした HAProxy 設定の例

```
# ...
listen api-server-6443
    bind *:6443
    mode tcp
      server master-00 192.168.83.89:6443 check inter 1s
      server master-01 192.168.84.90:6443 check inter 1s
      server master-02 192.168.85.99:6443 check inter 1s
      server bootstrap 192.168.80.89:6443 check inter 1s

listen machine-config-server-22623
    bind *:22623
    mode tcp
```

```
    server master-00 192.168.83.89:22623 check inter 1s
    server master-01 192.168.84.90:22623 check inter 1s
    server master-02 192.168.85.99:22623 check inter 1s
    server bootstrap 192.168.80.89:22623 check inter 1s

listen ingress-router-80
  bind *:80
  mode tcp
  balance source
    server worker-00 192.168.83.100:80 check inter 1s
    server worker-01 192.168.83.101:80 check inter 1s

listen ingress-router-443
  bind *:443
  mode tcp
  balance source
    server worker-00 192.168.83.100:443 check inter 1s
    server worker-01 192.168.83.101:443 check inter 1s

listen ironic-api-6385
  bind *:6385
  mode tcp
  balance source
    server master-00 192.168.83.89:6385 check inter 1s
    server master-01 192.168.84.90:6385 check inter 1s
    server master-02 192.168.85.99:6385 check inter 1s
    server bootstrap 192.168.80.89:6385 check inter 1s

listen inspector-api-5050
  bind *:5050
  mode tcp
  balance source
    server master-00 192.168.83.89:5050 check inter 1s
    server master-01 192.168.84.90:5050 check inter 1s
    server master-02 192.168.85.99:5050 check inter 1s
    server bootstrap 192.168.80.89:5050 check inter 1s
# ...
```

2. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーとそのリソースが動作していることを確認します。

   a. 次のコマンドを実行して応答を観察し、クラスターマシン設定 API が Kubernetes API サーバーリソースにアクセスできることを確認します。

      ```
      $ curl https://<loadbalancer_ip_address>:6443/version --insecure
      ```

      設定が正しい場合は、応答として JSON オブジェクトを受信します。

      ```
      {
        "major": "1",
        "minor": "11+",
        "gitVersion": "v1.11.0+ad103ed",
        "gitCommit": "ad103ed",
        "gitTreeState": "clean",
        "buildDate": "2019-01-09T06:44:10Z",
        "goVersion": "go1.10.3",
      ```

```
  "compiler": "gc",
  "platform": "linux/amd64"
}
```

b. 次のコマンドを実行して出力を確認し、クラスターマシン設定 API がマシン設定サーバーリソースからアクセスできることを確認します。

```
$ curl -v https://<loadbalancer_ip_address>:22623/healthz --insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
Content-Length: 0
```

c. 次のコマンドを実行して出力を確認し、コントローラーがポート 80 の Ingress Controller リソースにアクセスできることを確認します。

```
$ curl -I -L -H "Host: console-openshift-console.apps.<cluster_name>.<base_domain>"
http://<load_balancer_front_end_IP_address>
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.ocp4.private.opequon.net/
cache-control: no-cache
```

d. 次のコマンドを実行して出力を確認し、コントローラーがポート 443 の Ingress Controller リソースにアクセスできることを確認します。

```
$ curl -I -L --insecure --resolve console-openshift-console.apps.<cluster_name>.
<base_domain>:443:<Load Balancer Front End IP Address> https://console-openshift-
console.apps.<cluster_name>.<base_domain>
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

2. ユーザー管理ロードバランサーのフロントエンド IP アドレスをターゲットにするようにクラス

3. ユーザー管理ロードバランサーのフロントエンド IP アドレスをターゲットにするようにクラスターの DNS レコードを設定します。ロードバランサー経由で、クラスター API およびアプリケーションの DNS サーバーのレコードを更新する必要があります。

変更された DNS レコードの例

```
<load_balancer_ip_address>  A  api.<cluster_name>.<base_domain>
A record pointing to Load Balancer Front End
```

```
<load_balancer_ip_address>   A apps.<cluster_name>.<base_domain>
A record pointing to Load Balancer Front End
```

重要

DNS の伝播では、各 DNS レコードが使用可能になるまでに時間がかかる場合があります。各レコードを検証する前に、各 DNS レコードが伝播されることを確認してください。

4. OpenShift Container Platform クラスターでユーザー管理ロードバランサーを使用するには、クラスターの **install-config.yaml** ファイルで次の設定を指定する必要があります。

```
# ...
platform:
  vsphere:
    loadBalancer:
      type: UserManaged 1
      apiVIPs:
      - <api_ip> 2
      ingressVIPs:
      - <ingress_ip> 3
# ...
```

1. クラスターのユーザー管理ロードバランサーを指定するには、**type** パラメーターに **UserManaged** を設定します。パラメーターのデフォルトは **OpenShiftManagedDefault** で、これはデフォルトの内部ロードバランサーを示します。**openshift-kni-infra** namespace で定義されたサービスの場合、ユーザー管理ロードバランサーは **coredns** サービスをクラスター内の Pod にデプロイできますが、**keepalived** および **haproxy** サービスは無視します。

2. ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。Kubernetes API がユーザー管理ロードバランサーと通信できるように、ユーザー管理ロードバランサーのパブリック IP アドレスを指定します。

3. ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。ユーザー管理ロードバランサーのパブリック IP アドレスを指定して、ユーザー管理ロードバランサーがクラスターの Ingress トラフィックを管理できるようにします。

検証

1. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーと DNS レコード設定が動作していることを確認します。

   a. 次のコマンドを実行して出力を確認し、クラスター API にアクセスできることを確認しま

す。

```
$ curl https://api.<cluster_name>.<base_domain>:6443/version --insecure
```

設定が正しい場合は、応答として JSON オブジェクトを受信します。

```
{
  "major": "1",
  "minor": "11+",
  "gitVersion": "v1.11.0+ad103ed",
  "gitCommit": "ad103ed",
  "gitTreeState": "clean",
  "buildDate": "2019-01-09T06:44:10Z",
  "goVersion": "go1.10.3",
  "compiler": "gc",
  "platform": "linux/amd64"
  }
```

b. 次のコマンドを実行して出力を確認し、クラスターマシン設定にアクセスできることを確認します。

```
$ curl -v https://api.<cluster_name>.<base_domain>:22623/healthz --insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
Content-Length: 0
```

c. 以下のコマンドを実行して出力を確認し、ポートで各クラスターアプリケーションにアクセスできることを確認します。

```
$ curl http://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.<cluster-name>.<base domain>/
cache-control: no-cacheHTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=39HoZgztDnzjJkq/JuLJMeoKNXlfiVv2YgZc09c3TBOBU4Nl6kDXaJH1LdicNhN1UsQ
Wzon4Dor9GWGfopaTEQ==; Path=/; Secure
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Tue, 17 Nov 2020 08:42:10 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=9b714eb87e93cf34853e87a92d6894be; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

d. 次のコマンドを実行して出力を確認し、ポート 443 で各クラスターアプリケーションにアクセスできることを確認します。

```
$ curl https://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

23.2.5.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。
- オプション: クラスターを作成する前に、デフォルトのロードバランサーの代わりに外部ロードバランサーを設定します。

重要

インストールプログラムに API および Ingress 静的アドレスを指定する必要はありません。この設定を選択した場合は、追加のアクションを実行して、参照される各 vSphere サブネットから IP アドレスを受け入れるネットワークターゲットを定義する必要があります。「ユーザー管理ロードバランサーの設定」セクションを参照してください。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

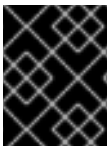
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

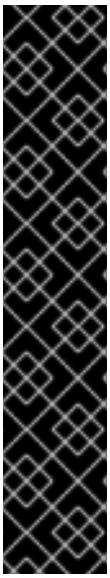


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

23.2.5.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

23.2.5.11. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

23.2.5.11.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

23.2.5.11.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

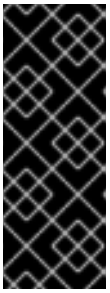
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

23.2.5.11.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

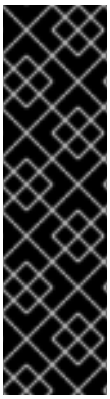
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Data Foundation など、クラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- "100Gi" の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリックストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

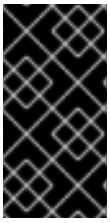
```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

23.2.5.11.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、**1** つのレプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
   2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
   3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
   4. 永続ボリューム要求 (PVC) のサイズ。

   b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1. カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

正しい PVC を参照するようにレジストリーストレージを設定する手順は、vSphere のレジストリーの設定 を参照してください。

23.2.5.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

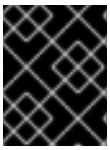
- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

23.2.5.13. コントロールプレーンで実行されるネットワークコンポーネントの設定

ネットワークコンポーネントは、コントロールプレーンノードでのみ実行するように設定できます。デフォルトで、OpenShift Container Platform はマシン設定プールの任意のノードが **ingressVIP** 仮想 IP アドレスをホストできるようにします。ただし、環境によっては、コントロールプレーンノードとは別のサブネットにコンピュートノードがデプロイされるため、コントロールプレーンノードで実行するために **ingressVIP** 仮想 IP アドレスを設定する必要があります。

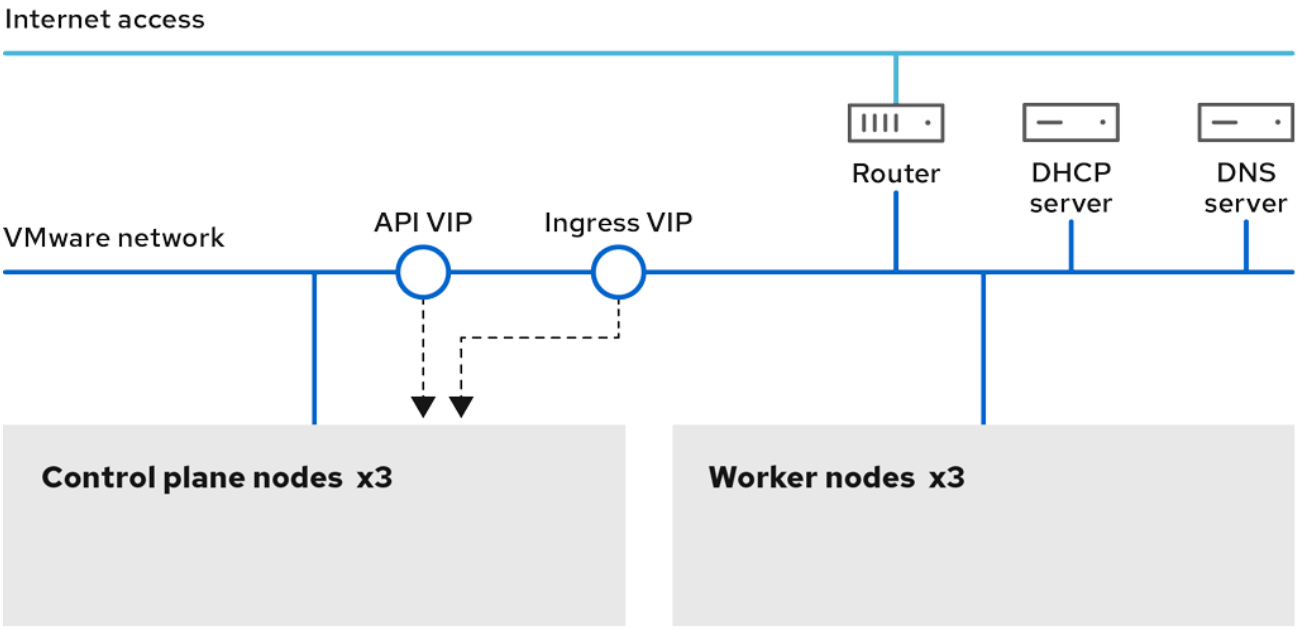
注記

別のサブネットにコンピュートマシンセットを作成することで、リモートノードをスケーリングできます。



重要

リモートノードを別々のサブネットにデプロイする場合は、コントロールプレーンノード専用に **ingressVIP** 仮想 IP アドレスを配置する必要があります。



手順

1. **install-config.yaml** ファイルを保存するディレクトリーに移動します。

```
$ cd ~/clusterconfigs
```

2. **manifests** サブディレクトリーに切り替えます。

```
$ cd manifests
```

3. **cluster-network-avoid-workers-99-config.yaml** という名前のファイルを作成します。

```
$ touch cluster-network-avoid-workers-99-config.yaml
```

4. エディターで **cluster-network-avoid-workers-99-config.yaml** ファイルを開き、Operator 設定を記述するカスタムリソース (CR) を入力します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 50-worker-fix-ipi-rwn
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/kubernetes/manifests/keepalived.yaml
          mode: 0644
          contents:
            source: data:,
```

このマニフェストは、 **ingressVIP** 仮想 IP アドレスをコントロールプレーンノードに配置します。また、このマニフェストは、コントロールプレーンノードにのみ以下のプロセスをデプロイします。

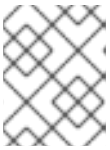
- **openshift-ingress-operator**
- **keepalived**

5. **cluster-network-avoid-workers-99-config.yaml** ファイルを保存します。
6. **manifests/cluster-ingress-default-ingresscontroller.yaml** ファイルを作成します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/master: ""
```

7. **manifests** ディレクトリーのバックアップの作成を検討してください。インストーラーは、クラスターの作成時に **manifests**/ ディレクトリーを削除します。
8. **cluster-scheduler-02-config.yml** マニフェストを変更し、**mastersSchedulable** フィールドを **true** に設定して、コントロールプレーンノードをスケジュール対象にします。デフォルトでは、コントロールプレーンノードはスケジュール対象ではありません。以下に例を示します。

```
$ sed -i "s;mastersSchedulable: false;mastersSchedulable: true;g"
clusterconfigs/manifests/cluster-scheduler-02-config.yml
```



注記

この手順の完了後にコントロールプレーンノードをスケジュールできない場合には、クラスターのデプロイに失敗します。

23.2.5.14. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

23.2.6. ネットワークが制限された環境での vSphere へのクラスターのインストール

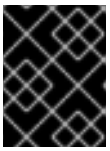
OpenShift Container Platform 4.16 では、インストールリリースコンテンツの内部ミラーを作成して、制限されたネットワーク内の VMware vSphere インフラストラクチャーにクラスターをインストールできます。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

23.2.6.1. 前提条件

- installer-provisioned infrastructure を使用したクラスターのインストールの準備 のタスクを完了した。
- VMware プラットフォームのライセンスを確認した。Red Hat は VMware ライセンスに制限を設けていませんが、一部の VMware インフラストラクチャーコンポーネントにはライセンスが必要です。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターの 永続ストレージ をプロビジョニングした。プライベートイメージレジストリーをデプロイするには、ストレージで ReadWriteMany アクセスモードを指定する必要があります。
- OpenShift Container Platform インストーラーは、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

23.2.6.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベ

アメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

23.2.6.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

23.2.6.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

23.2.6.4. ネットワークが制限されたインストール用の RHCOS イメージの作成

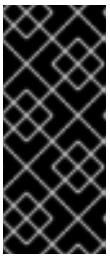
Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードし、OpenShift Container Platform をネットワークが制限された VMware vSphere 環境にインストールします。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、プログラムはミラーレジストリーホスト上に置かれます。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、RHEL 8 用の OpenShift Container Platform 4.16 の最新リリースを選択します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - vSphere**イメージをダウンロードします。
4. ダウンロードしたイメージを、bastion サーバーからアクセス可能な場所にアップロードします。

これで、イメージが制限されたインストールで利用可能になります。OpenShift Container Platform デプロイメントで使用するイメージの名前または場所をメモします。

23.2.6.5. VMware vSphere のリージョンとゾーンの有効化

OpenShift Container Platform クラスターを、単一の VMware vCenter で実行される複数の vSphere データセンターにデプロイできます。各データセンターは複数のクラスターを実行できます。この設定により、クラスターの障害を引き起こす可能性のあるハードウェア障害やネットワーク停止のリスクが軽減されます。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

重要

VMware vSphere のリージョンおよびゾーンの有効化機能には、クラスター内のデフォルトのストレージドライバーとして vSphere Container Storage Interface (CSI) ドライバーが必要です。そのため、この機能は新しくインストールされたクラスターでのみ使用できます。

以前のリリースからアップグレードされたクラスターの場合は、クラスターの CSI 自動移行を有効にする必要があります。その後、アップグレードされたクラスターに対して複数のリージョンとゾーンを設定できます。

デフォルトのインストール設定では、クラスターが単一の vSphere データセンターにデプロイされます。クラスターを複数の vSphere データセンターにデプロイする場合は、リージョンおよびゾーン機能を有効にするインストール設定ファイルを作成する必要があります。

デフォルトの **install-config.yaml** ファイルには **vcenters フィールド** と **FailureDomains** フィールドが含まれており、OpenShift Container Platform クラスターに複数の vSphere データセンターとクラスターを指定できます。単一のデータセンターで設定される vSphere 環境に OpenShift Container Platform クラスターをインストールする場合は、これらのフィールドを空白のままにすることができます。

次のリストでは、クラスターのゾーンとリージョンの定義に関連する用語について説明します。

- 障害ドメイン: リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。
- リージョン: vCenter データセンターを指定します。リージョンを定義するには、**openshift-region** タグカテゴリーのタグを使用します。

- ゾーン: vCenter クラスターを指定します。ゾーンを定義するには、**openshift-zone** タグカテゴリーのタグを使用します。

注記

install-config.yaml ファイルで複数の障害ドメインを指定する予定がある場合は、設定ファイルを作成する前に、タグカテゴリー、ゾーンタグ、およびリージョンタグを作成する必要があります。

リージョンを表す vCenter データセンターごとに vCenter タグを作成する必要があります。さらに、データセンターで実行されるクラスターごとに、ゾーンを表す vCenter タグを作成する必要があります。タグを作成した後、各タグをそれぞれのデータセンターとクラスターにアタッチする必要があります。

次の表は、単一の VMware vCenter で実行されている複数の vSphere データセンターを含む設定のリージョン、ゾーン、タグ間の関係の例を示しています。

データセンター (リージョン)	クラスター (ゾーン)	タグ
米国東部	us-east-1	us-east-1a
		us-east-1b
	us-east-2	us-east-2a
		us-east-2b
us-west	us-west-1	us-west-1a
		us-west-1b
	us-west-2	us-west-2a
		us-west-2b

関連情報

- 追加の VMware vSphere 設定パラメーター
- 非推奨の VMware vSphere 設定パラメーター
- vSphere の自動移行
- VMware vSphere CSI ドライバー Operator

23.2.6.6. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値がある。
- ミラーレジストリーの証明書の内容を取得している。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、アクセス可能な場所にアップロードしました。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   ディレクトリーを指定する場合:

   - ディレクトリーに **execute** 権限があることを確認します。この権限は、インストールディレクトリーで Terraform バイナリーを実行するために必要です。
   - 空のディレクトリーを使用します。ブートストラップ X.509 証明書などの一部のインストールアセットは有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。
      iii. vCenter インスタンスの名前を指定します。
      iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
      インストールプログラムは vCenter インスタンスに接続します。

v. 接続する vCenter インスタンスのデータセンターを選択します。

注記

インストール設定ファイルを作成した後、そのファイルを変更して複数の vSphere データセンター環境を作成できます。これは、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできることを意味します。この環境の作成の詳細については、**VMware vSphere のリージョンとゾーンの有効化** を参照してください。

vi. 使用するデフォルトの vCenter データストアを選択します。

警告

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion を使用する Storage Distributed Resource Scheduler (SDRS) がデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage DRS を無効にする必要があります。

複数のデータストアパスを指定することはできません。複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。
入力するクラスター名は、DNS レコードの設定時に指定したクラスター名と一致する必要があります。

2. **install-config.yaml** ファイルで **platform.vsphere.clusterOSImage** の値をイメージの場所または名前に設定します。以下に例を示します。

```
platform:
  vsphere:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
vmware.x86_64.ova?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

   a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

   ```
   pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
   "you@example.com"}}}'
   ```

   <mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

   b. **additionalTrustBundle** パラメーターおよび値を追加します。

   ```
   additionalTrustBundle: |
     -----BEGIN CERTIFICATE-----

   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
     -----END CERTIFICATE-----
   ```

   この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。証明書ファイルは、既存の信頼できる認証局、またはミラーレジストリー用に生成した自己署名証明書のいずれかです。

   c. 次の YAML の抜粋のようなイメージコンテンツリソースを追加します。

   ```
   imageContentSources:
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: quay.io/openshift-release-dev/ocp-release
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: registry.redhat.io/ocp/release
   ```

   これらの値には、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

   d. オプション: パブリッシュストラテジーを **Internal** に設定します。

   ```
   publish: Internal
   ```

   このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。
   パラメーターの詳細については、インストール設定パラメーターを参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- インストール設定パラメーター

23.2.6.6.1. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- architecture: amd64
  name:  <worker_node>
  platform: {}
  replicas: 3
controlPlane: (3)
  architecture: amd64
  name: <parent_node>
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: test (4)
platform:
  vsphere: (5)
    apiVIPs:
      - 10.0.0.1
    failureDomains: (6)
    - name: <failure_domain_name>
      region: <default_region_name>
      server: <fully_qualified_domain_name>
      topology:
        computeCluster: "/<datacenter>/host/<cluster>"
        datacenter: <datacenter>
        datastore: "/<datacenter>/datastore/<datastore>" (7)
        networks:
        - <VM_Network_name>
        resourcePool: "/<datacenter>/host/<cluster>/Resources/<resourcePool>" (8)
        folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>"
        tagIDs: (9)
        - <tag_id> (10)
      zone: <default_zone_name>
    ingressVIPs:
    - 10.0.0.2
    vcenters:
    - datacenters:
```

```
      - <datacenter>
      password: <password>
      port: 443
      server: <fully_qualified_domain_name>
      user: administrator@vsphere.local
    diskType: thin (11)
    clusterOSImage: http://mirror.example.com/images/rhcos-47.83.202103221318-0-
vmware.x86_64.ova (12)
fips: false
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (13)
sshKey: 'ssh-ed25519 AAAA...'
additionalTrustBundle: | (14)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (15)
- mirrors:
  - <mirror_host_name>:<mirror_port>/<repo_name>/release
  source: <source_image_1>
- mirrors:
  - <mirror_host_name>:<mirror_port>/<repo_name>/release-images
  source: <source_image_2>
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (3) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(4) DNS レコードに指定したクラスター名。

(5) オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

(6) リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。

(7) 仮想マシンファイル、テンプレート、ISO イメージを保持する vSphere データストアへのパス。

重要

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion はデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage vMotion を無効にする必要があります。

複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

8 オプション: マシン作成用の既存のリソースプールを提供します。値を指定しない場合、インストールプログラムは vSphere クラスターのルートリソースプールを使用します。

9 オプション: OpenShift Container Platform によって作成された各仮想マシンには、クラスターに固有の一意のタグが割り当てられます。割り当てられたタグにより、クラスターの使用停止時に、関連付けられた仮想マシンをインストールプログラムが識別して削除できるようになります。インストールプログラムによってプロビジョニングされた仮想マシンに割り当てる追加のタグ ID を最大 10 個までリストできます。

10 インストールプログラムによって関連付けられるタグの ID。たとえば、**urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL** です。タグ ID の決定の詳細は、vSphere Tags and Attributes ドキュメントを参照してください。

11 vSphere ディスクのプロビジョニング方法。

12 bastion サーバーからアクセス可能な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所。

13 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

14 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

15 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

注記

OpenShift Container Platform 4.12 以降では、**apiVIP** および **ingressVIP** 設定は非推奨です。代わりに、リスト形式を使用して、**apiVIPs** および **ingressVIPs** 設定に値を入力します。

23.2.6.6.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

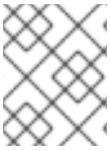
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

23.2.6.6.3. VMware vCenter のリージョンとゾーンの設定

デフォルトのインストール設定ファイルを変更して、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできるようにします。

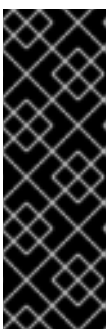
OpenShift Container Platform の以前のリリースのデフォルトの **install-config.yaml** ファイル設定は非推奨になりました。非推奨のデフォルト設定を引き続き使用できますが、**openshift-installer** により、設定ファイル内の非推奨のフィールドの使用を示す警告メッセージが表示されます。

重要

この例では、**govc** コマンドを使用します。**govc** コマンドは、VMware から入手できるオープンソースコマンドです。Red Hat からは入手できません。Red Hat サポートチームは **govc** コマンドを保守していません。**govc** のダウンロードとインストールの手順については、VMware ドキュメント Web サイトを参照してください。

前提条件

- 既存の **install-config.yaml** インストール設定ファイルがあります。



重要

VMware vCenter Server のデータセンターオブジェクトをプロビジョニングできるように、OpenShift Container Platform クラスターに少なくとも 1 つの障害ドメインを指定する必要があります。異なるデータセンター、クラスター、データストア、その他のコンポーネントに仮想マシンノードをプロビジョニングする必要がある場合は、複数の障害ドメインを指定することを検討してください。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

手順

1. 次の **govc** コマンドラインツールコマンドを入力して、**openshift-region** および **openshift-zone** vCenter タグカテゴリーを作成します。

重要

openshift-region および **openshift-zone** vCenter タグカテゴリーに異なる名前を指定すると、OpenShift Container Platform クラスターのインストールは失敗します。

```
$ govc tags.category.create -d "OpenShift region" openshift-region
```

```
$ govc tags.category.create -d "OpenShift zone" openshift-zone
```

2. クラスターをデプロイする各リージョン vSphere データセンターのリージョンタグを作成するには、ターミナルで次のコマンドを入力します。

```
$ govc tags.create -c <region_tag_category> <region_tag>
```

3. クラスターをデプロイする vSphere クラスターごとにゾーンタグを作成するには、次のコマンドを入力します。

```
$ govc tags.create -c <zone_tag_category> <zone_tag>
```

4. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにリージョンタグをアタッチします。

```
$ govc tags.attach -c <region_tag_category> <region_tag_1> /<datacenter_1>
```

5. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにゾーンタグをアタッチします。

```
$ govc tags.attach -c <zone_tag_category> <zone_tag_1> /<datacenter_1>/host/vcs-mdcnc-workload-1
```

6. インストールプログラムが含まれるディレクトリーに移動し、選択したインストール要件に従ってクラスターデプロイメントを初期化します。

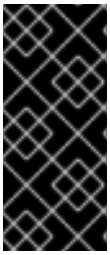
vSphere センターで定義された複数のデータセンターを含むサンプル install-config.yaml ファイル

```
---
compute:
---
  vsphere:
     zones:
       - "<machine_pool_zone_1>"
       - "<machine_pool_zone_2>"
---
controlPlane:
---
```

```
vsphere:
      zones:
        - "<machine_pool_zone_1>"
        - "<machine_pool_zone_2>"
---
platform:
  vsphere:
    vcenters:
---
    datacenters:
      - <datacenter1_name>
      - <datacenter2_name>
    failureDomains:
    - name: <machine_pool_zone_1>
      region: <region_tag_1>
      zone: <zone_tag_1>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter1>
        computeCluster: "/<datacenter1>/host/<cluster1>"
        networks:
        - <VM_Network1_name>
        datastore: "/<datacenter1>/datastore/<datastore1>"
        resourcePool: "/<datacenter1>/host/<cluster1>/Resources/<resourcePool1>"
        folder: "/<datacenter1>/vm/<folder1>"
    - name: <machine_pool_zone_2>
      region: <region_tag_2>
      zone: <zone_tag_2>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter2>
        computeCluster: "/<datacenter2>/host/<cluster2>"
        networks:
        - <VM_Network2_name>
        datastore: "/<datacenter2>/datastore/<datastore2>"
        resourcePool: "/<datacenter2>/host/<cluster2>/Resources/<resourcePool2>"
        folder: "/<datacenter2>/vm/<folder2>"
---
```

23.2.6.7. ユーザー管理ロードバランサーのサービス

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。



重要

ユーザー管理ロードバランサーの設定は、ベンダーのロードバランサーによって異なります。

このセクションの情報と例は、ガイドラインのみを目的としています。ベンダーのロードバランサーに関する詳細は、ベンダーのドキュメントを参照してください。

Red Hat は、ユーザー管理ロードバランサーに対して次のサービスをサポートしています。

- Ingress Controller

- OpenShift API
- OpenShift MachineConfig API

ユーザー管理ロードバランサーに対して、これらのサービスの 1 つを設定するか、すべてを設定するかを選択できます。一般的な設定オプションは、Ingress Controller サービスのみを設定することです。次の図は、各サービスの詳細を示しています。

図23.7 OpenShift Container Platform 環境で動作する Ingress Controller を示すネットワークワークフローの例

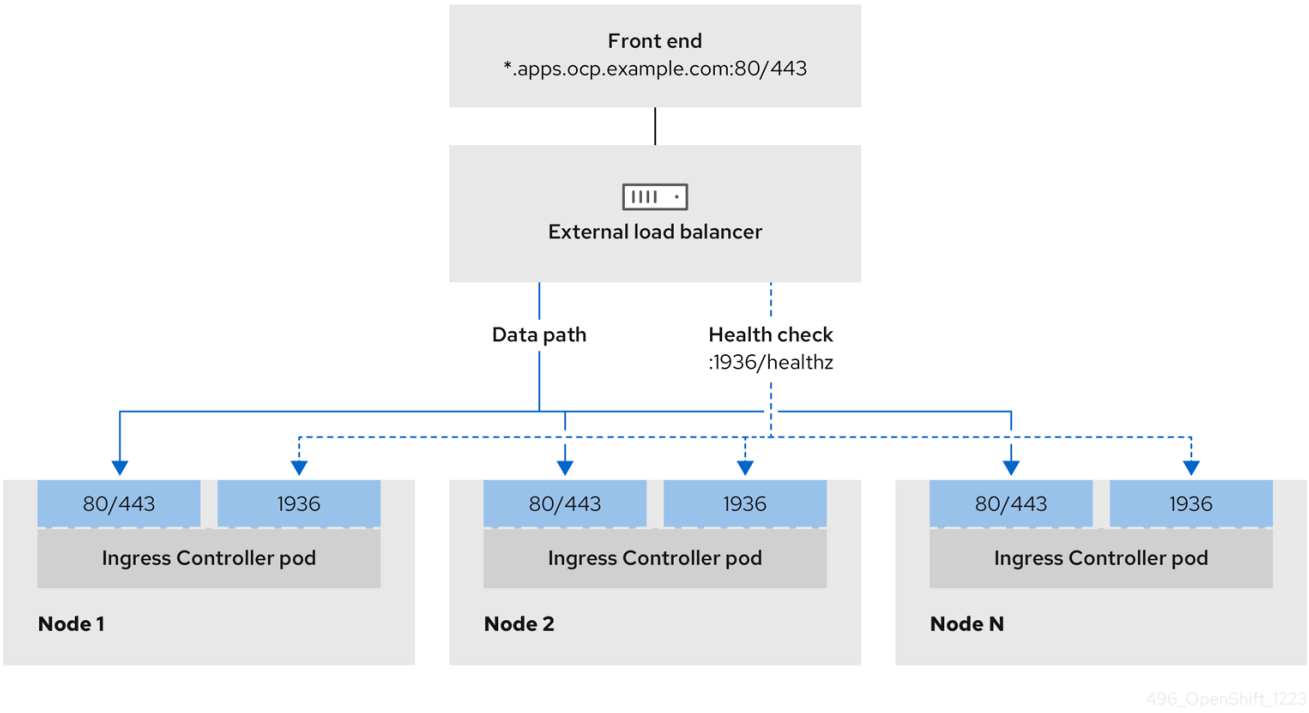


図23.8 OpenShift Container Platform 環境で動作する OpenShift API を示すネットワークワークフローの例

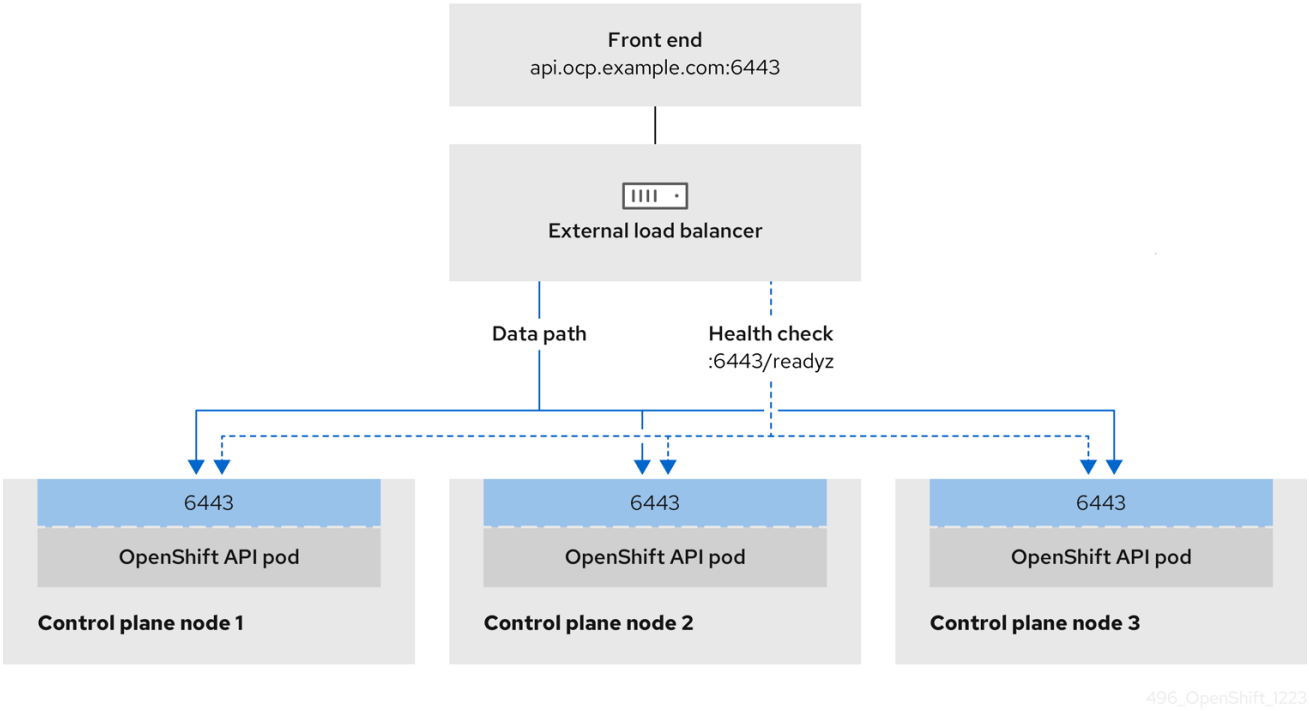
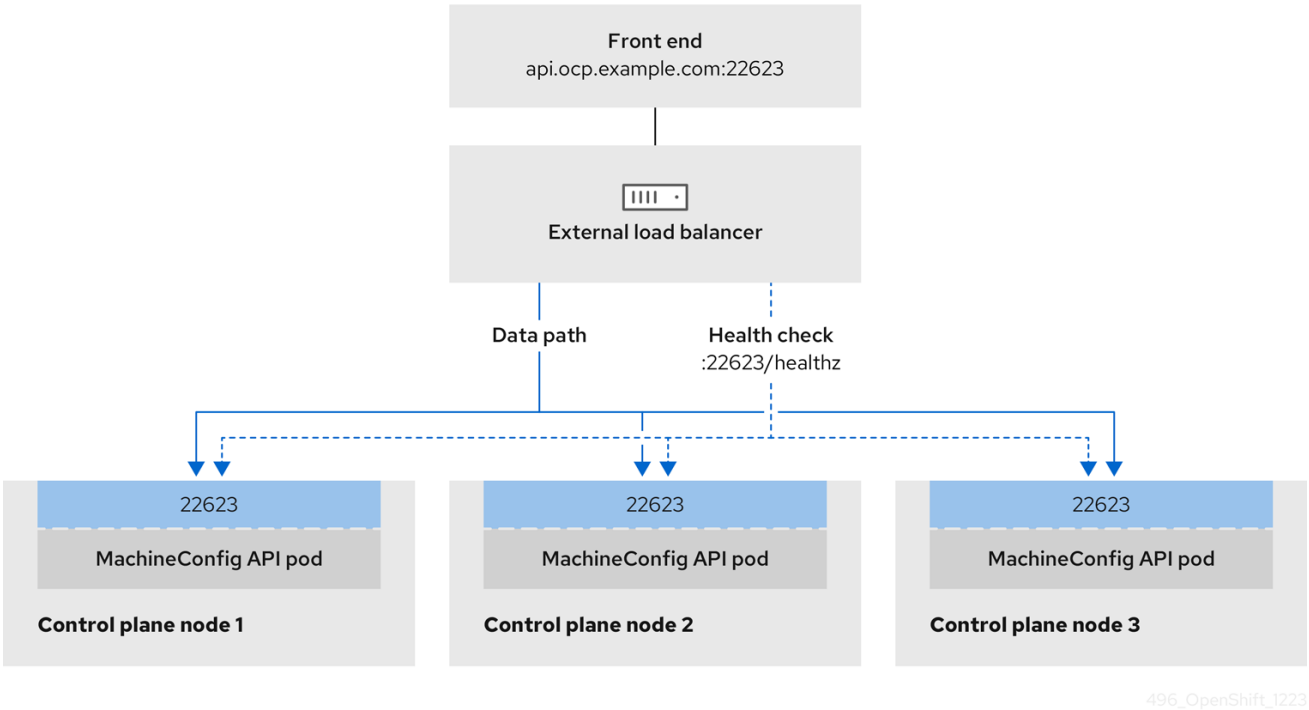


図23.9 OpenShift Container Platform 環境で動作する OpenShift MachineConfig API を示すネットワークワークフローの例



ユーザー管理ロードバランサーでは、次の設定オプションがサポートされています。

- ノードセレクターを使用して、Ingress Controller を特定のノードのセットにマッピングします。このセットの各ノードに静的 IP アドレスを割り当てるか、Dynamic Host Configuration Protocol (DHCP) から同じ IP アドレスを受け取るように各ノードを設定する必要があります。インフラストラクチャーノードは通常、このタイプの設定を受け取ります。

- サブネット上のすべての IP アドレスをターゲットにします。この設定では、ロードバランサーターゲットを再設定せずにネットワーク内でノードを作成および破棄できるため、メンテナンスオーバーヘッドを削減できます。**/27** や **/28** などの小規模なネットワーク上に設定されたマシンを使用して Ingress Pod をデプロイする場合、ロードバランサーのターゲットを簡素化できます。

  ヒント

  マシン config プールのリソースを確認することで、ネットワーク内に存在するすべての IP アドレスをリスト表示できます。

OpenShift Container Platform クラスターのユーザー管理ロードバランサーを設定する前に、以下の情報を考慮してください。

- フロントエンド IP アドレスの場合、フロントエンド IP アドレス、Ingress Controller のロードバランサー、および API ロードバランサーに同じ IP アドレスを使用できます。この機能については、ベンダーのドキュメントを確認してください。

- バックエンド IP アドレスの場合、ユーザー管理ロードバランサーの有効期間中に OpenShift Container Platform コントロールプレーンノードの IP アドレスが変更されないことを確認します。次のいずれかのアクションを実行すると、これを実現できます。

  - 各コントロールプレーンノードに静的 IP アドレスを割り当てます。

  - ノードが DHCP リースを要求するたびに、DHCP から同じ IP アドレスを受信するように各ノードを設定します。ベンダーによっては、DHCP リースは IP 予約または静的 DHCP 割り当ての形式になる場合があります。

- Ingress Controller バックエンドサービスのユーザー管理ロードバランサーで Ingress Controller を実行する各ノードを手動で定義します。たとえば、Ingress Controller が未定義のノードに移動すると、接続が停止する可能性があります。

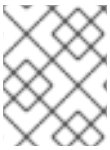
23.2.6.7.1. ユーザー管理ロードバランサーの設定

デフォルトのロードバランサーの代わりに、ユーザーが管理するロードバランサーを使用するように OpenShift Container Platform クラスターを設定できます。

重要

ユーザー管理ロードバランサーを設定する前に、ユーザー管理ロードバランサーのサービス" セクションを必ずお読みください。

ユーザー管理ロードバランサー用に設定するサービスに適用される次の前提条件をお読みください。



注記

クラスター上で実行される MetalLB は、ユーザー管理ロードバランサーとして機能します。

OpenShift API の前提条件

- フロントエンド IP アドレスを定義している。

- TCP ポート 6443 および 22623 は、ロードバランサーのフロントエンド IP アドレスで公開されている。以下の項目を確認します。
  - ポート 6443 が OpenShift API サービスにアクセスできる。
  - ポート 22623 が Ignition 起動設定をノードに提供できる。
- フロントエンド IP アドレスとポート 6443 へは、OpenShift Container Platform クラスターの外部の場所にいるシステムのすべてのユーザーがアクセスできる。
- フロントエンド IP アドレスとポート 22623 は、OpenShift Container Platform ノードからのみ到達できる。
- ロードバランサーバックエンドは、ポート 6443 および 22623 の OpenShift Container Platform コントロールプレーンノードと通信できる。

Ingress Controller の前提条件

- フロントエンド IP アドレスを定義している。
- TCP ポート 443 および 80 はロードバランサーのフロントエンド IP アドレスで公開されている。
- フロントエンドの IP アドレス、ポート 80、ポート 443 へは、OpenShift Container Platform クラスターの外部の場所にあるシステムの全ユーザーがアクセスできる。
- フロントエンドの IP アドレス、ポート 80、ポート 443 は、OpenShift Container Platform クラスターで動作するすべてのノードから到達できる。
- ロードバランサーバックエンドは、ポート 80、443、および 1936 で Ingress Controller を実行する OpenShift Container Platform ノードと通信できる。

ヘルスチェック URL 仕様の前提条件

ほとんどのロードバランサーは、サービスが使用可能か使用不可かを判断するヘルスチェック URL を指定して設定できまうｓ．OpenShift Container Platform は、OpenShift API、Machine Configuration API、および Ingress Controller バックエンドサービスのこれらのヘルスチェックを提供します。

次の例は、前にリスト表示したバックエンドサービスのヘルスチェック仕様を示しています。

Kubernetes API ヘルスチェック仕様の例

```
Path: HTTPS:6443/readyz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Machine Config API ヘルスチェック仕様の例

```
Path: HTTPS:22623/healthz
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 10
Interval: 10
```

Ingress Controller のヘルスチェック仕様の例

```
Path: HTTP:1936/healthz/ready
Healthy threshold: 2
Unhealthy threshold: 2
Timeout: 5
Interval: 10
```

手順

1. HAProxy Ingress Controller を設定して、ポート 6443、22623、443、および 80 でロードバランサーからクラスターへのアクセスを有効化できるようにします。必要に応じて、HAProxy 設定で単一のサブネットの IP アドレスまたは複数のサブネットの IP アドレスを指定できます。

   #### 1つのサブネットをリストした HAProxy 設定の例

   ```
   # ...
   listen my-cluster-api-6443
       bind 192.168.1.100:6443
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /readyz
     http-check expect status 200
       server my-cluster-master-2 192.168.1.101:6443 check inter 10s rise 2 fall 2
       server my-cluster-master-0 192.168.1.102:6443 check inter 10s rise 2 fall 2
       server my-cluster-master-1 192.168.1.103:6443 check inter 10s rise 2 fall 2

   listen my-cluster-machine-config-api-22623
       bind 192.168.1.100:22623
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /healthz
     http-check expect status 200
       server my-cluster-master-2 192.168.1.101:22623 check inter 10s rise 2 fall 2
       server my-cluster-master-0 192.168.1.102:22623 check inter 10s rise 2 fall 2
       server my-cluster-master-1 192.168.1.103:22623 check inter 10s rise 2 fall 2

   listen my-cluster-apps-443
       bind 192.168.1.100:443
       mode tcp
       balance roundrobin
     option httpchk
     http-check connect
     http-check send meth GET uri /healthz/ready
     http-check expect status 200
       server my-cluster-worker-0 192.168.1.111:443 check port 1936 inter 10s rise 2 fall 2
       server my-cluster-worker-1 192.168.1.112:443 check port 1936 inter 10s rise 2 fall 2
       server my-cluster-worker-2 192.168.1.113:443 check port 1936 inter 10s rise 2 fall 2

   listen my-cluster-apps-80
     bind 192.168.1.100:80
   ```

```
    mode tcp
    balance roundrobin
   option httpchk
   http-check connect
   http-check send meth GET uri /healthz/ready
   http-check expect status 200
     server my-cluster-worker-0 192.168.1.111:80 check port 1936 inter 10s rise 2 fall 2
     server my-cluster-worker-1 192.168.1.112:80 check port 1936 inter 10s rise 2 fall 2
     server my-cluster-worker-2 192.168.1.113:80 check port 1936 inter 10s rise 2 fall 2
# ...
```

複数のサブネットをリストした HAProxy 設定の例

```
# ...
listen api-server-6443
    bind *:6443
    mode tcp
      server master-00 192.168.83.89:6443 check inter 1s
      server master-01 192.168.84.90:6443 check inter 1s
      server master-02 192.168.85.99:6443 check inter 1s
      server bootstrap 192.168.80.89:6443 check inter 1s

listen machine-config-server-22623
    bind *:22623
    mode tcp
      server master-00 192.168.83.89:22623 check inter 1s
      server master-01 192.168.84.90:22623 check inter 1s
      server master-02 192.168.85.99:22623 check inter 1s
      server bootstrap 192.168.80.89:22623 check inter 1s

listen ingress-router-80
    bind *:80
    mode tcp
    balance source
      server worker-00 192.168.83.100:80 check inter 1s
      server worker-01 192.168.83.101:80 check inter 1s

listen ingress-router-443
    bind *:443
    mode tcp
    balance source
      server worker-00 192.168.83.100:443 check inter 1s
      server worker-01 192.168.83.101:443 check inter 1s

listen ironic-api-6385
    bind *:6385
    mode tcp
    balance source
      server master-00 192.168.83.89:6385 check inter 1s
      server master-01 192.168.84.90:6385 check inter 1s
      server master-02 192.168.85.99:6385 check inter 1s
      server bootstrap 192.168.80.89:6385 check inter 1s

listen inspector-api-5050
    bind *:5050
    mode tcp
```

```
    balance source
      server master-00 192.168.83.89:5050 check inter 1s
      server master-01 192.168.84.90:5050 check inter 1s
      server master-02 192.168.85.99:5050 check inter 1s
      server bootstrap 192.168.80.89:5050 check inter 1s
# ...
```

2. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーとそのリソースが動作していることを確認します。

   a. 次のコマンドを実行して応答を観察し、クラスターマシン設定 API が Kubernetes API サーバーリソースにアクセスできることを確認します。

      ```
      $ curl https://<loadbalancer_ip_address>:6443/version --insecure
      ```

      設定が正しい場合は、応答として JSON オブジェクトを受信します。

      ```
      {
        "major": "1",
        "minor": "11+",
        "gitVersion": "v1.11.0+ad103ed",
        "gitCommit": "ad103ed",
        "gitTreeState": "clean",
        "buildDate": "2019-01-09T06:44:10Z",
        "goVersion": "go1.10.3",
        "compiler": "gc",
        "platform": "linux/amd64"
      }
      ```

   b. 次のコマンドを実行して出力を確認し、クラスターマシン設定 API がマシン設定サーバーリソースからアクセスできることを確認します。

      ```
      $ curl -v https://<loadbalancer_ip_address>:22623/healthz --insecure
      ```

      設定が正しい場合、コマンドの出力には次の応答が表示されます。

      ```
      HTTP/1.1 200 OK
      Content-Length: 0
      ```

   c. 次のコマンドを実行して出力を確認し、コントローラーがポート 80 の Ingress Controller リソースにアクセスできることを確認します。

      ```
      $ curl -I -L -H "Host: console-openshift-console.apps.<cluster_name>.<base_domain>"
      http://<load_balancer_front_end_IP_address>
      ```

      設定が正しい場合、コマンドの出力には次の応答が表示されます。

      ```
      HTTP/1.1 302 Found
      content-length: 0
      location: https://console-openshift-console.apps.ocp4.private.opequon.net/
      cache-control: no-cache
      ```

   d. 次のコマンドを実行して出力を確認し、コントローラーがポート 443 の Ingress Controller リソースにアクセスできることを確認します。

```
$ curl -I -L --insecure --resolve console-openshift-console.apps.<cluster_name>.
<base_domain>:443:<Load Balancer Front End IP Address> https://console-openshift-
console.apps.<cluster_name>.<base_domain>
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dG
LgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

3. ユーザー管理ロードバランサーのフロントエンド IP アドレスをターゲットにするようにクラスターの DNS レコードを設定します。ロードバランサー経由で、クラスター API およびアプリケーションの DNS サーバーのレコードを更新する必要があります。

変更された DNS レコードの例

```
<load_balancer_ip_address>  A  api.<cluster_name>.<base_domain>
A record pointing to Load Balancer Front End
```

```
<load_balancer_ip_address>   A apps.<cluster_name>.<base_domain>
A record pointing to Load Balancer Front End
```

重要

DNS の伝播では、各 DNS レコードが使用可能になるまでに時間がかかる場合があります。各レコードを検証する前に、各 DNS レコードが伝播されることを確認してください。

4. OpenShift Container Platform クラスターでユーザー管理ロードバランサーを使用するには、クラスターの **install-config.yaml** ファイルで次の設定を指定する必要があります。

```
# ...
platform:
  vsphere:
    loadBalancer:
      type: UserManaged 1
      apiVIPs:
      - <api_ip> 2
```

```
    ingressVIPs:
    - <ingress_ip> 3
# ...
```

1. クラスターのユーザー管理ロードバランサーを指定するには、**type** パラメーターに **UserManaged** を設定します。パラメーターのデフォルトは **OpenShiftManagedDefault** で、これはデフォルトの内部ロードバランサーを示します。**openshift-kni-infra** namespace で定義されたサービスの場合、ユーザー管理ロードバランサーは **coredns** サービスをクラスター内の Pod にデプロイできますが、**keepalived** および **haproxy** サービスは無視します。

2. ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。Kubernetes API がユーザー管理ロードバランサーと通信できるように、ユーザー管理ロードバランサーのパブリック IP アドレスを指定します。

3. ユーザー管理ロードバランサーを指定する場合に必須のパラメーターです。ユーザー管理ロードバランサーのパブリック IP アドレスを指定して、ユーザー管理ロードバランサーがクラスターの Ingress トラフィックを管理できるようにします。

検証

1. **curl** CLI コマンドを使用して、ユーザー管理ロードバランサーと DNS レコード設定が動作していることを確認します。

   a. 次のコマンドを実行して出力を確認し、クラスター API にアクセスできることを確認します。

   ```
   $ curl https://api.<cluster_name>.<base_domain>:6443/version --insecure
   ```

   設定が正しい場合は、応答として JSON オブジェクトを受信します。

   ```
   {
     "major": "1",
     "minor": "11+",
     "gitVersion": "v1.11.0+ad103ed",
     "gitCommit": "ad103ed",
     "gitTreeState": "clean",
     "buildDate": "2019-01-09T06:44:10Z",
     "goVersion": "go1.10.3",
     "compiler": "gc",
     "platform": "linux/amd64"
     }
   ```

   b. 次のコマンドを実行して出力を確認し、クラスターマシン設定にアクセスできることを確認します。

   ```
   $ curl -v https://api.<cluster_name>.<base_domain>:22623/healthz --insecure
   ```

   設定が正しい場合、コマンドの出力には次の応答が表示されます。

   ```
   HTTP/1.1 200 OK
   Content-Length: 0
   ```

c. 以下のコマンドを実行して出力を確認し、ポートで各クラスターアプリケーションにアクセスできることを確認します。

```
$ curl http://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.<cluster-name>.<base domain>/
cache-control: no-cacheHTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-token=39HoZgztDnzjJkq/JuLJMeoKNXlfiVv2YgZc09c3TBOBU4NI6kDXaJH1LdicNhN1UsQWzon4Dor9GWGfopaTEQ==; Path=/; Secure
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Tue, 17 Nov 2020 08:42:10 GMT
content-type: text/html; charset=utf-8
set-cookie: 1e2670d92730b515ce3a1bb65da45062=9b714eb87e93cf34853e87a92d6894be; path=/; HttpOnly; Secure; SameSite=None
cache-control: private
```

d. 次のコマンドを実行して出力を確認し、ポート 443 で各クラスターアプリケーションにアクセスできることを確認します。

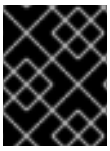
```
$ curl https://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --insecure
```

設定が正しい場合、コマンドの出力には次の応答が表示されます。

```
HTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-token=UlYWOyQ62LWjw2h003xtYSKlh1a0Py2hhctw0WmV2YEdhJjFyQwWcGBsja261dGLgaYO0nxzVErhiXt6QepA7g==; Path=/; Secure; SameSite=Lax
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Wed, 04 Oct 2023 16:29:38 GMT
content-type: text/html; charset=utf-8
set-cookie: 1e2670d92730b515ce3a1bb65da45062=1bf5e9573c9a2760c964ed1659cc1673; path=/; HttpOnly; Secure; SameSite=None
cache-control: private
```

23.2.6.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。
- ホスト上のクラウドプロバイダーアカウントに、クラスターをデプロイするための適切な権限があることが確認されました。アカウントの権限が正しくないと、インストールプロセスが失敗し、不足している権限を示すエラーメッセージが表示されます。
- オプション: クラスターを作成する前に、デフォルトのロードバランサーの代わりに外部ロードバランサーを設定します。

重要

インストールプログラムに API および Ingress 静的アドレスを指定する必要はありません。この設定を選択した場合は、追加のアクションを実行して、参照される各 vSphere サブネットから IP アドレスを受け入れるネットワークターゲットを定義する必要があります。「ユーザー管理ロードバランサーの設定」セクションを参照してください。

手順

- インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

検証

クラスターのデプロイが正常に完了すると、次のようになります。

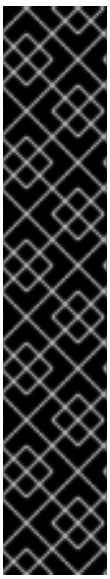
- ターミナルには、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報など、クラスターにアクセスするための指示が表示されます。
- 認証情報は **<installation_directory>/.openshift_install.log** にも出力されます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

23.2.6.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

23.2.6.10. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

23.2.6.11. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

23.2.6.11.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

23.2.6.11.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

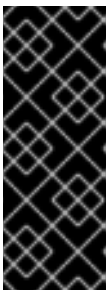
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

23.2.6.11.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

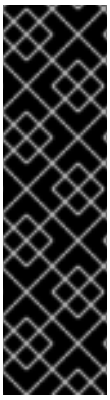
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Data Foundation など、クラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- "100Gi" の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリックストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

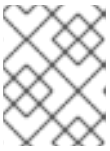
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME           VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                               True        False         False      6h50m
```

23.2.6.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

23.2.6.13. 次のステップ

- クラスターのカスタマイズ

- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

23.3. USER-PROVISIONED INFRASTRUCTURE

23.3.1. user-provisioned infrastructure の vSphere インストール要件

プロビジョニングしたインフラストラクチャーへのインストールを開始する前に、vSphere 環境が次のインストール要件を満たしていることを確認してください。

23.3.1.1. VMware vSphere インフラストラクチャーの要件

OpenShift Container Platform クラスターは、使用するコンポーネントの要件に合わせて、以下に示す VMware vSphere インスタンスのいずれかのバージョンにインストールする必要があります。

- バージョン 7.0 Update 2 以降
- バージョン 8.0 Update 1 以降

これらのリリースは、どちらも Container Storage Interface (CSI) の移行をサポートしています。CSI の移行は、OpenShift Container Platform 4.16 ではデフォルトで有効になっています。

VMware vSphere インフラストラクチャーは、オンプレミスまたは次の表に示す要件を満たす VMware Cloud Verified プロバイダー でホストできます。

表23.18 vSphere 仮想環境のバージョン要件

仮想環境製品	必須バージョン
VMware 仮想ハードウェア	15 以降
vSphere ESXi ホスト	7.0 Update 2 以降、8.0 Update 1 以降
vCenter ホスト	7.0 Update 2 以降、8.0 Update 1 以降

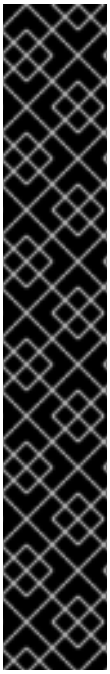
重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

表23.19 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	仮想ハードウェアバージョン 15 を搭載した vSphere 7.0 Update 2 (以降) または vSphere 8.0 Update 1 (以降)	このハイパーバイザーのバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。RHCOS と互換性のある Red Hat Enterprise Linux (RHEL) の最新バージョンでサポートされているハードウェアの詳細は、Red Hat Customer Portal の ハードウェア を参照してください。
オプション: Networking(NSX-T)	vSphere 7.0 Update 2 以降、vSphere 8.0 Update 1 以降	OpenShift Container Platform には、vSphere 7.0 Update 2 以降または vSphere 8.0 Update 1 以降が必要です。NSX および OpenShift Container Platform の互換性についての詳細は、VMware の NSX コンテナープラグインドキュメント の リリースノート セクションを参照してください。
CPU マイクロアーキテクチャー	x86-64-v2 以降	OpenShift 4.13 以降では、マイクロアーキテクチャーの要件が x86-64-v2 に発生する RHEL 9.2 ホストオペレーティングシステムをベースにしています。RHEL マイクロアーキテクチャー要件 に関するドキュメント を参照してください。このナレッジベースの記事 に記載されている手順に従って、互換性を確認できます。



重要

Oracle® Cloud Infrastructure (OCI) および Oracle® Cloud VMware Solution (OCVS) サービス上で動作するクラスターワークロードの最適なパフォーマンス条件を確保するには、ブロックボリュームのボリュームパフォーマンスユニット (VPU) がワークロードに合わせてサイズ設定されていることを確認してください。

次のリストを参考に、特定のパフォーマンスニーズに応じて必要な VPU を選択してください。

- テストまたは概念実証環境: 100 GB、20 - 30 VPU。
- 基本実稼働環境: 500 GB、60 VPU。
- 頻繁に使用される実稼働環境: 500 GB 以上、100 以上の VPU。

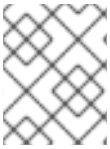
更新とスケーリングアクティビティーに十分な容量を提供するために、追加の VPU を割り当てることを検討してください。Block Volume Performance Levels (Oracle ドキュメント) を参照してください。

23.3.1.2. VMware vSphere CSI Driver Operator の要件

vSphere Container Storage Interface (CSI) Driver Operator をインストールするには、次の要件を満たす必要があります。

- VMware vSphere バージョン: 7.0 Update 2 以降、8.0 Update 1 以降
- vCenter バージョン: 7.0 Update 2 以降、8.0 Update 1 以降
- ハードウェアバージョン 15 以降の仮想マシン
- クラスターにサードパーティーの vSphere CSI ドライバーがインストールされていない

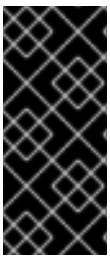
サードパーティーの vSphere CSI ドライバーがクラスターに存在する場合、OpenShift Container Platform はそれを上書きしません。サードパーティーの vSphere CSI ドライバーが存在すると、OpenShift Container Platform を OpenShift Container Platform 4.13 以降にアップグレードできなくなります。



注記

VMware vSphere CSI Driver Operator は、インストールマニフェストの **platform: vsphere** でデプロイされたクラスターでのみサポートされます。

Container Storage Interface (CSI) ドライバー、vSphere CSI Driver Operator、および vSphere Problem Detector Operator のカスタムロールを作成できます。カスタムロールには、各 vSphere オブジェクトに最小限の権限セットを割り当てる権限セットを含めることができます。つまり、CSI ドライバー、vSphere CSI Driver Operator、および vSphere Problem Detector Operator はこれらのオブジェクトとの基本的な対話を確立できます。



重要

vCenter への OpenShift Container Platform クラスターのインストールは、「必要な vCenter アカウントの特権」セクションで説明されているすべての特権のリストに対してテストされています。このすべての特権のリストに準拠することで、制限された特権セットを持つカスタムロールの作成時に予期しない動作やサポートされていない動作が発生する可能性を抑制できます。

関連情報

- サードパーティーの vSphere CSI ドライバーを削除する場合は、サードパーティーの vSphere CSI ドライバーの削除 を参照してください。
- vSphere ノードのハードウェアバージョンを更新する場合は、vSphere で稼働するノードのハードウェア更新 を参照してください。
- ストレージコンポーネントの最小権限

23.3.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

23.3.1.3.1. vCenter の要件

指定のインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、vSphere アカウントに必要なリソースの読み取りと作成のための権限が含まれている必要があります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

例23.8 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	常時	**Cns.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.Update** **StorageProfile.View**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	常時	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere ポートグループ	常時	**Network.Assign**
仮想マシンフォルダー	常時	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.EditDevice** **VirtualMachine.Config.Memory** **VirtualMachine.Config.RemoveDisk** **VirtualMachine.Config.Rename** **Host.Config.Storage** **VirtualMachine.Config.ResetGuestInfo** **VirtualMachine.Config.Resource** **VirtualMachine.Config.Settings** **VirtualMachine.Config.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Provisioning.Clone **VirtualMachine.Provisioning.MarkAsTemplate** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。「Machine API の最小権限」の表を参照してください。	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.EditDevice** **VirtualMachine.Config.Memory** **VirtualMachine.Config.RemoveDisk** **VirtualMachine.Config.Rename** **VirtualMachine.Config.ResetGuestInfo** **VirtualMachine.Config.Resource** **VirtualMachine.Config.Settings** **VirtualMachine.Config.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

例23.9 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	常時	Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere Datastore	常時	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	常時	**Network."Assign network"**
仮想マシンフォルダー	常時	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Cr**	"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
	eate 権限と VirtualMachine.Inventory.Delete 権限は任意です。	disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new"

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例23.10 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	必要になる場合	子への伝播	パーミッションが必要
vSphere vCenter	常時	False	必要な特権がリスト表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権がリスト表示
vSphere vCenter Cluster	既存のリソースプール	False	**ReadOnly** パーミッション
	クラスタールートの仮想マシン	True	必要な特権がリスト表示
vSphere vCenter Datastore	常時	False	必要な特権がリスト表示
vSphere Switch	常時	False	**ReadOnly** パーミッション
vSphere ポートグループ	常時	False	必要な特権がリスト表示

vSphere オブジェクト	必要になる場合	子への伝播	パーミッションが必要
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権がリスト表示
vSphere vCenter リソースプール	既存のリソースプール	True	必要な特権がリスト表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

最低限必要な vCenter アカウントの特権
カスタムロールを作成してそのロールに特権を割り当てた後、特定の vSphere オブジェクトを選択し、オブジェクトごとにカスタムロールをユーザーまたはグループに割り当てることで権限を作成できます。

vSphere オブジェクトの権限を作成するか、権限の作成を要求する前に、vSphere オブジェクトに適用する最小限の権限を決定してください。このタスクを実行すると、vSphere オブジェクトと OpenShift Container Platform アーキテクチャーの間の基本的な対話を確立できます。

重要

カスタムロールを作成し、そのロールに特権を割り当てなかった場合、vSphere Server はデフォルトで **Read Only** ロールをそのカスタムロールに割り当てます。クラウドプロバイダー API の場合、カスタムロールは **Read Only** ロールの特権を継承するだけで済むことに注意してください。

グローバル管理者の特権を持つアカウントがニーズに合わない場合は、カスタムロールの作成を検討してください。

重要

必要な特権が設定されていないアカウントはサポートされません。vCenter への OpenShift Container Platform クラスターのインストールは、「必要な vCenter アカウントの特権」セクションで説明されているすべての特権のリストに対してテストされています。このすべての権限のリストに準拠することで、制限された特権セットを持つカスタムロールの作成時に予期しない動作が発生する可能性を抑制できます。

以下の表に、特定の OpenShift Container Platform アーキテクチャーと対話する vSphere オブジェクトの最小権限のリストを示します。

例23.11 コンポーネントのインストール後の管理のための最小権限

ロールの vSphere オブジェクト	必要になる場合	必要な特権

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	常時	**Cns.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.Update** **StorageProfile.View**
vSphere vCenter Cluster	クラスタールートに仮想マシンを作成する場合	**Host.Config.Storage** **Resource.AssignVMToPool**
vSphere vCenter リソースプール	**install-config.yaml** ファイルに既存のリソースプールを指定する場合	**Host.Config.Storage**
vSphere Datastore	常時	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**
vSphere ポートグループ	常時	**Network.Assign**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
仮想マシンフォルダー	常時	**VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.Memory** **VirtualMachine.Config.Settings** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。クラスターで Machine API を使用しており、API の最小権限セットを設定する必要がある場合は、「Machine API の最小権限」の表を参照してください。	**Resource.AssignVMToPool** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Provisioning.DeployTemplate**

例23.12 ストレージコンポーネントの最小権限

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	常時	**Cns.Searchable** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **StorageProfile.Update** **StorageProfile.View**
vSphere vCenter Cluster	クラスタールートに仮想マシンを作成する場合	**Host.Config.Storage**
vSphere vCenter リソースプール	**install-config.yaml** ファイルに既存のリソースプールを指定する場合	**Host.Config.Storage**
vSphere Datastore	常時	**Datastore.Browse** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**
vSphere ポートグループ	常時	**Read-Only**
仮想マシンフォルダー	常時	**VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddRemoveDevice**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。クラスターで Machine API を使用しており、API の最小権限セットを設定する必要がある場合は、「Machine API の最小権限」の表を参照してください。	**VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddRemoveDevice**

例23.13 Machine API の最小権限

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	常時	**InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.Update StorageProfile.View**
vSphere vCenter Cluster	クラスタールートに仮想マシンを作成する場合	**Resource.AssignVMToPool**
vSphere vCenter リソースプール	**install-config.yaml** ファイルに既存のリソースプールを指定する場合	**Read-Only**
vSphere Datastore	常時	**Datastore.AllocateSpace Datastore.Browse**
vSphere ポートグループ	常時	**Network.Assign**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
仮想マシンフォルダー	常時	**VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.Memory** **VirtualMachine.Config.Settings** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合。user-provisioned infrastructure の場合、クラスターで Machine API を使用しないのであれば、**VirtualMachine.Inventory.Create** 権限と **VirtualMachine.Inventory.Delete** 権限は任意です。	**Resource.AssignVMToPool** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Provisioning.DeployTemplate**

OpenShift Container Platform と vMotion の使用

vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュート専用の vMotion をサポートします。これは、**一般に**、vMotion に関するすべての VMware ベストプラクティスを満たすことを意味します。
  コンピュートプレーンノードとコントロールプレーンノードの稼働時間を確保するには、vMotion に関する VMware のベストプラクティスに従い、VMware のアンチアフィニティールールを使用して、メンテナンスまたはハードウェアの問題時の OpenShift Container Platform の可用性を向上させます。

vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。Pod で vSphere ボリュームを使用している場合、手動または Storage vMotion を介してデータストア間で VM を移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生し、データ損失が発生する可能性があります。

- OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

重要

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion を使用する Storage Distributed Resource Scheduler (SDRS) がデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage DRS を無効にする必要があります。

複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

クラスターリソース
提供したインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合は、vCenter インスタンスに以下のリソースを作成する必要があります。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に

利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに動的ホスト設定プロトコル (DHCP) を使用し、クラスターマシンに永続的な IP アドレスを提供するように DHCP サーバーが設定されていることを確認します。

注記

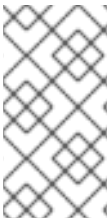
静的 IP アドレスを使用してノードをプロビジョニングする場合は、ネットワークに DHCP を使用する必要はありません。

DHCP サーバーを使用するには、デフォルトゲートウェイを設定します。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。

ネットワークに動的ホスト設定プロトコル (DHCP) を使用し、クラスターマシンに永続的な IP アドレスを提供するように DHCP サーバーが設定されていることを確認する必要があります。DHCP リースでは、デフォルトゲートウェイを使用するように DHCP を設定する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。

制限された環境にインストールする場合、制限されたネットワーク内の仮想マシンは、ノード、永続ボリュームクレーム (PVC)、およびその他のリソースをプロビジョニングおよび管理するために、vCenter にアクセスできる必要があります。

さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。



注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表23.20 必要な DNS レコード

コンポーネント	レコード	説明

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME (Canonical Name) レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

関連情報

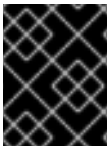
- vSphere でコンピュートマシンセットを作成する

23.3.1.3.2. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表23.21 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

23.3.1.3.3. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表23.22 最小リソース要件

マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [1]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [2]	2	8 GB	100 GB	300

1. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

2. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

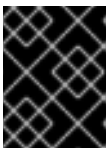
関連情報

- ストレージの最適化

23.3.1.3.4. 仮想マシンを暗号化するための要件

以下の要件を満たすと、OpenShift Container Platform 4.16 をインストールする前に、仮想マシンを暗号化できます。

- vSphere で標準キープロバイダーを設定しました。詳細については、vCenter Server への KMS の追加 を参照してください。

  重要

  vCenter のネイティブキープロバイダーはサポートされていません。詳細については、vSphere Native Key Provider の概要 を参照してください。

- クラスターをホスティングしているすべての ESXi ホストでホスト暗号化モードを有効にしました。詳細については、ホスト暗号化モードの有効化 を参照してください。
- すべての暗号化権限が有効になっている vSphere アカウントがあります。詳細については、暗号化操作の権限 を参照してください。

「RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始」セクションで OVF テンプレートをデプロイする場合は、OVF テンプレートのストレージを選択する際、Encrypt this virtual machine オプションを選択します。クラスターのインストールが完了したら、仮想マシンの暗号化に使用した暗号化ストレージポリシーを使用するストレージクラスを作成します。

関連情報

- 暗号化されたストレージクラスの作成

23.3.1.3.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管

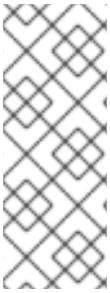
理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

23.3.1.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。



注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

23.3.1.3.6.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

23.3.1.3.6.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表23.23 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表23.24 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表23.25 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

Ethernet アダプターのハードウェアアドレス要件
クラスターの仮想マシンをプロビジョニングする場合、各仮想マシンに設定されたイーサネットインターフェイスは VMware Organizationally Unique Identifier (OUI) 割り当て範囲から MAC アドレスを使用する必要があります。

- **00:05:69:00:00:00** - **00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00** - **00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00** - **00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00** to **00:50:56:3F:FF:FF**

VMware OUI 外の MAC アドレスが使用される場合、クラスターのインストールは成功しません。

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

23.3.1.3.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表23.26 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。

コンポーネント	レコード	説明
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

23.3.1.3.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例23.14 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
```

```
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例23.15 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1. Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。
2. Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。
3. ブートストラップマシンの逆引き DNS 解決を提供します。
4. 5. 6. コントロールプレーンマシンの逆引き DNS 解決を提供します。
7. 8. コンピュートマシンの逆引き DNS 解決を提供します。

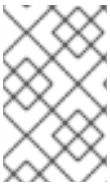
注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

23.3.1.3.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーイン

フラストラクチャーを分離してスケーリングすることができます。



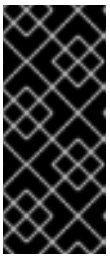
注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

    - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
    - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表23.27 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

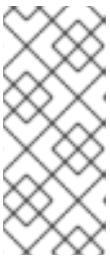
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表23.28 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

23.3.1.3.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューショ

ンを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例23.16 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
```

```
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 5
    bind *:443
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:443 check inter 1s
    server compute1 compute1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 6
    bind *:80
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:80 check inter 1s
    server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

23.3.2. user-provisioned infrastructure を使用したクラスターのインストールの準備

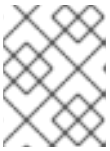
以下の手順を実行して、vSphere に OpenShift Container Platform クラスターをインストールする準備をします。

- インストールプログラムをダウンロードします。

注記

非接続環境にインストールする場合は、ミラーリングしたコンテンツからインストールプログラムを抽出します。詳細は、非接続インストール用のイメージのミラーリング を参照してください。

- OpenShift CLI (**oc**) をインストールします。



注記

非接続環境にインストールする場合は、ミラーホストに **oc** をインストールします。

- SSH キーペアを生成します。OpenShift Container Platform クラスターのデプロイ後にこのキーペアを使用して、クラスターのノードに対する認証を行うことができます。
- user-provisioned infrastructure を準備します。
- DNS 解決を検証します。

23.3.2.1. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

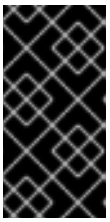
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

23.3.2.2. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

23.3.2.3. クラスターノードの SSH アクセス用のキーペアの生成

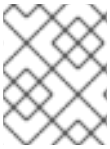
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの ~/**.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

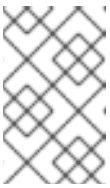
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

23.3.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

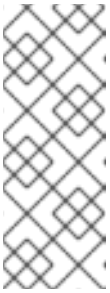
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

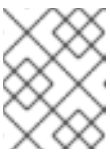
1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。
   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。



注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。



注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプ**

ロビジョニングされるインフラストラクチャーのネットワーク要件のセクションを参照してください。

重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

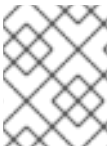
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

23.3.2.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   注記

   出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

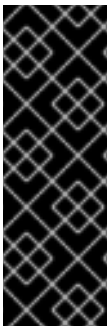
c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

23.3.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した vSphere へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、独自にプロビジョニングする VMware vSphere インフラストラクチャーに、クラスターをインストールできます。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

23.3.3.1. 前提条件

- user-provisioned infrastructure を使用したクラスターのインストールの準備 のタスクを完了した。
- VMware プラットフォームのライセンスを確認した。Red Hat は VMware ライセンスに制限を設けていませんが、一部の VMware インフラストラクチャーコンポーネントにはライセンスが必要です。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- クラスターの 永続ストレージ をプロビジョニングした。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- インストールを完了するには、vSphere ホストに Red Hat Enterprise Linux CoreOS(RHCOS) OVA をアップロードする必要があります。このプロセスを完了するマシンには、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。

- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

プロキシーを設定する場合は、このサイトリストも確認してください。

23.3.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

23.3.3.3. VMware vSphere のリージョンとゾーンの有効化

OpenShift Container Platform クラスターを、単一の VMware vCenter で実行される複数の vSphere データセンターにデプロイできます。各データセンターは複数のクラスターを実行できます。この設定により、クラスターの障害を引き起こす可能性のあるハードウェア障害やネットワーク停止のリスクが軽減されます。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

重要

VMware vSphere のリージョンおよびゾーンの有効化機能には、クラスター内のデフォルトのストレージドライバーとして vSphere Container Storage Interface (CSI) ドライバーが必要です。そのため、この機能は新しくインストールされたクラスターでのみ使用できます。

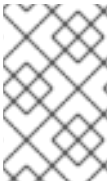
以前のリリースからアップグレードされたクラスターの場合は、クラスターの CSI 自動移行を有効にする必要があります。その後、アップグレードされたクラスターに対して複数のリージョンとゾーンを設定できます。

デフォルトのインストール設定では、クラスターが単一の vSphere データセンターにデプロイされます。クラスターを複数の vSphere データセンターにデプロイする場合は、リージョンおよびゾーン機能を有効にするインストール設定ファイルを作成する必要があります。

デフォルトの **install-config.yaml** ファイルには **vcenters フィールド** と **FailureDomains** フィールドが含まれており、OpenShift Container Platform クラスターに複数の vSphere データセンターとクラスターを指定できます。単一のデータセンターで設定される vSphere 環境に OpenShift Container Platform クラスターをインストールする場合は、これらのフィールドを空白のままにすることができます。

次のリストでは、クラスターのゾーンとリージョンの定義に関連する用語について説明します。

- 障害ドメイン: リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。
- リージョン: vCenter データセンターを指定します。リージョンを定義するには、**openshift-region** タグカテゴリーのタグを使用します。
- ゾーン: vCenter クラスターを指定します。ゾーンを定義するには、**openshift-zone** タグカテゴリーのタグを使用します。



注記

install-config.yaml ファイルで複数の障害ドメインを指定する予定がある場合は、設定ファイルを作成する前に、タグカテゴリー、ゾーンタグ、およびリージョンタグを作成する必要があります。

リージョンを表す vCenter データセンターごとに vCenter タグを作成する必要があります。さらに、データセンターで実行されるクラスターごとに、ゾーンを表す vCenter タグを作成する必要があります。タグを作成した後、各タグをそれぞれのデータセンターとクラスターにアタッチする必要があります。

次の表は、単一の VMware vCenter で実行されている複数の vSphere データセンターを含む設定のリージョン、ゾーン、タグ間の関係の例を示しています。

データセンター (リージョン)	クラスター (ゾーン)	タグ
米国東部	us-east-1	us-east-1a
		us-east-1b
	us-east-2	us-east-2a
		us-east-2b
us-west	us-west-1	us-west-1a
		us-west-1b
	us-west-2	us-west-2a

データセンター (リージョン)	クラスター (ゾーン)	タグ
		us-west-2b

関連情報

- 追加の VMware vSphere 設定パラメーター
- 非推奨の VMware vSphere 設定パラメーター
- vSphere の自動移行
- VMware vSphere CSI ドライバー Operator

23.3.3.4. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

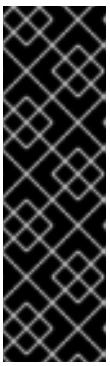
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

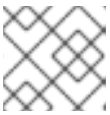
```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



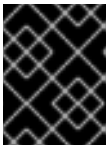
注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. 3 ノードクラスターをインストールする場合は、**compute.replicas** パラメーターを **0** に設定し

て、**install-config.yaml** ファイルを変更します。これにより、クラスターのコントロールプレーンがスケジュール可能になります。詳細については、「vSphere への 3 ノードクラスターのインストール」を参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- インストール設定パラメーター

23.3.3.4.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
additionalTrustBundlePolicy: Proxyonly
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- architecture: amd64
  name: <worker_node>
  platform: {}
  replicas: 0 (3)
controlPlane: (4)
  architecture: amd64
  name: <parent_node>
  platform: {}
  replicas: 3 (5)
metadata:
  creationTimestamp: null
  name: test (6)
networking:
---
platform:
  vsphere:
    failureDomains: (7)
    - name: <failure_domain_name>
      region: <default_region_name>
      server: <fully_qualified_domain_name>
      topology:
        computeCluster: "/<datacenter>/host/<cluster>"
        datacenter: <datacenter> (8)
        datastore: "/<datacenter>/datastore/<datastore>" (9)
        networks:
        - <VM_Network_name>
        resourcePool: "/<datacenter>/host/<cluster>/Resources/<resourcePool>" (10)
        folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (11)
      zone: <default_zone_name>
```

```
    vcenters:
    - datacenters:
      - <datacenter>
      password: <password> 12
      port: 443
      server: <fully_qualified_domain_name> 13
      user: administrator@vsphere.local
    diskType: thin 14
fips: false 15
pullSecret: '{"auths": ...}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 4 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。両方のセクションで単一のマシンプールが定義されるため、使用されるコントロールプレーンは 1 つだけです。OpenShift Container Platform は、複数のコンピューティングプールの定義をサポートしていません。

3 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

5 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

6 DNS レコードに指定したクラスター名。

7 リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。

8 vSphere データセンター。

9 仮想マシンファイル、テンプレート、ISO イメージを保持する vSphere データストアへのパス。

重要

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion はデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage vMotion を無効にする必要があります。

複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

10 オプション: インストーラーによってプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存のリソースプールの絶対パス (例:

11 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供していて、**thin** という名前のデフォルトの **StorageClass** オブジェクトを使用したくない場合は、**install-config.yaml** ファイルから **folder** パラメーターを省略できます。

12 vSphere ユーザーに関連付けられたパスワード。

13 vCenter サーバーの完全修飾ホスト名または IP アドレス。

重要

Cloud Controller Manager Operator は、指定されたホスト名または IP アドレスに対して接続チェックを行います。到達可能な vCenter サーバーに対して、ホスト名または IP アドレスを指定していることを確認してください。存在しない vCenter サーバーにメタデータを提供すると、クラスターのインストールはブートストラップ段階で失敗します。

14 vSphere ディスクのプロビジョニング方法。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

16 OpenShift Cluster Manager から取得したプルシークレット。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

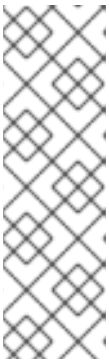
17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

23.3.3.4.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

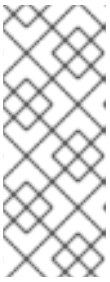
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名され

ない限り必要になります。

5 オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



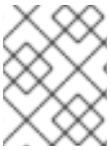
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

23.3.3.4.3. VMware vCenter のリージョンとゾーンの設定

デフォルトのインストール設定ファイルを変更して、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできるようにします。

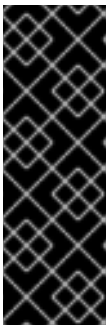
OpenShift Container Platform の以前のリリースのデフォルトの **install-config.yaml** ファイル設定は非推奨になりました。非推奨のデフォルト設定を引き続き使用できますが、**openshift-installer** により、設定ファイル内の非推奨のフィールドの使用を示す警告メッセージが表示されます。

重要

この例では、**govc** コマンドを使用します。**govc** コマンドは、VMware から入手できるオープンソースコマンドです。Red Hat からは入手できません。Red Hat サポートチームは **govc** コマンドを保守していません。**govc** のダウンロードとインストールの手順については、VMware ドキュメント Web サイトを参照してください。

前提条件

- 既存の **install-config.yaml** インストール設定ファイルがあります。

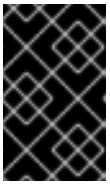


重要

VMware vCenter Server のデータセンターオブジェクトをプロビジョニングできるように、OpenShift Container Platform クラスターに少なくとも 1 つの障害ドメインを指定する必要があります。異なるデータセンター、クラスター、データストア、その他のコンポーネントに仮想マシンノードをプロビジョニングする必要がある場合は、複数の障害ドメインを指定することを検討してください。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

手順

1. 次の **govc** コマンドラインツールコマンドを入力して、**openshift-region** および **openshift-zone** vCenter タグカテゴリーを作成します。



重要

openshift-region および **openshift-zone** vCenter タグカテゴリーに異なる名前を指定すると、OpenShift Container Platform クラスターのインストールは失敗します。

```
$ govc tags.category.create -d "OpenShift region" openshift-region
```

```
$ govc tags.category.create -d "OpenShift zone" openshift-zone
```

2. クラスターをデプロイする各リージョン vSphere データセンターのリージョンタグを作成するには、ターミナルで次のコマンドを入力します。

```
$ govc tags.create -c <region_tag_category> <region_tag>
```

3. クラスターをデプロイする vSphere クラスターごとにゾーンタグを作成するには、次のコマンドを入力します。

```
$ govc tags.create -c <zone_tag_category> <zone_tag>
```

4. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにリージョンタグをアタッチします。

```
$ govc tags.attach -c <region_tag_category> <region_tag_1> /<datacenter_1>
```

5. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにゾーンタグをアタッチします。

```
$ govc tags.attach -c <zone_tag_category> <zone_tag_1> /<datacenter_1>/host/vcs-mdcnc-workload-1
```

6. インストールプログラムが含まれるディレクトリーに移動し、選択したインストール要件に従ってクラスターデプロイメントを初期化します。

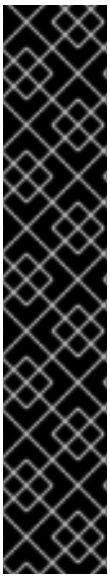
vSphere センターで定義された複数のデータセンターを含むサンプル install-config.yaml ファイル

```
---
compute:
---
  vsphere:
      zones:
        - "<machine_pool_zone_1>"
        - "<machine_pool_zone_2>"
---
controlPlane:
---
vsphere:
      zones:
        - "<machine_pool_zone_1>"
        - "<machine_pool_zone_2>"
---
platform:
  vsphere:
    vcenters:
---
    datacenters:
      - <datacenter1_name>
      - <datacenter2_name>
    failureDomains:
    - name: <machine_pool_zone_1>
      region: <region_tag_1>
      zone: <zone_tag_1>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter1>
        computeCluster: "/<datacenter1>/host/<cluster1>"
        networks:
        - <VM_Network1_name>
        datastore: "/<datacenter1>/datastore/<datastore1>"
        resourcePool: "/<datacenter1>/host/<cluster1>/Resources/<resourcePool1>"
        folder: "/<datacenter1>/vm/<folder1>"
    - name: <machine_pool_zone_2>
      region: <region_tag_2>
      zone: <zone_tag_2>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter2>
        computeCluster: "/<datacenter2>/host/<cluster2>"
        networks:
        - <VM_Network2_name>
        datastore: "/<datacenter2>/datastore/<datastore2>"
        resourcePool: "/<datacenter2>/host/<cluster2>/Resources/<resourcePool2>"
        folder: "/<datacenter2>/vm/<folder2>"
---
```

23.3.3.5. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシン、コンピュートマシンセット、およびコントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- コンピュートマシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

23.3.3.6. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware vSphere でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

23.3.3.7. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。
- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
2. ブートストラップノードの以下の二次的な Ignition 設定ファイルを、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

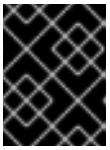
ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。
   - **<installation_directory>/master.ign**
   - **<installation_directory>/worker.ign**
   - **<installation_directory>/merge-bootstrap.ign**
4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
   たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

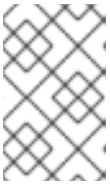
5. RHCOS OVA イメージを取得します。イメージは、RHCOS イメージミラー ページから入手できます。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設定に適した場所にあるストレージを動的にプロビジョニングします。
7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。



注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。
   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。
   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。
   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。
   e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。
      - ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
      - **install-config.yaml** ファイルで指定したデータストアを選択します。
      - 仮想マシンを暗号化する場合は、**Encrypt this virtual machine** を選択します。詳細については、「仮想マシンを暗号化するための要件」セクションを参照してください。
   f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

OVF テンプレートの作成時には、**Customize template** タブで値を指定しないでテンプ

g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。

重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをコンピュートマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. 必要に応じて、仮想マシンテンプレートで設定された仮想ハードウェアバージョンを更新します。詳細は、VMware ドキュメントの Upgrading a virtual machine to the latest hardware version を参照してください。

重要

必要に応じて、仮想マシンを作成する前に、仮想マシンテンプレートのハードウェアバージョンをバージョン 15 に更新することが推奨されます。vSphere で実行しているクラスターノード用にハードウェアバージョン 13 を使用することは非推奨となりました。インポートしたテンプレートがハードウェアバージョン 13 にデフォルト設定されている場合は、仮想マシンテンプレートをハードウェアバージョン 15 にアップグレードする前に、ESXi ホストが 6.7U3 以降を使用していることを確認する必要があります。vSphere のバージョンが 6.7U3 未満の場合は、このアップグレード手順を省略できます。ただし、OpenShift Container Platform の今後のバージョンでは、ハードウェアバージョン 13 および vSphere バージョンのサポートが 6.7U3 未満になる予定です。

9. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

注記

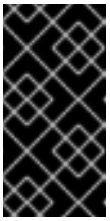
vSphere インストール全体のすべての仮想マシン名が一意であることを確認してください。

c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

e. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。

f. **Customize hardware** タブで、**Advanced Parameters** をクリックします。



重要

次の設定の提案は、例としてのみ使用されます。クラスター管理者は、クラスターに課せられるリソース需要に従ってリソースを設定する必要があります。クラスターリソースを最適に管理するには、クラスターのルートリソースプールからリソースプールを作成することを検討してください。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。
  - 静的 IP 設定を行います。

    コマンドの例

    ```
    $ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none
    nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
    ```

    コマンドの例

    ```
    $ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
    nameserver=8.8.8.8"
    ```

  - vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

    コマンドの例

    ```
    $ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-
    kargs=${IPCFG}"
    ```

- **Attribute** フィールドおよび **Values** フィールドにデータを指定して、以下の設定パラメーター名と値を追加します。作成するパラメーターごとに **Add** ボタンを選択してください。
  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。
  - クラスターの root リソースプールから子リソースプールを作成します。この子リソースプールでリソースの割り当てを実行します。

g. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

h. 残りの設定手順を完了します。**Finish** ボタンをクリックして、クローン作成操作を完了します。

i. **Virtual Machines** タブで仮想マシンを右クリックし、**Power → Power On** を選択します。

j. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

次のステップ

- 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

23.3.3.8. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

vSphere テンプレートを OpenShift Container Platform クラスターにデプロイした後に、そのクラスター内のマシンの仮想マシン (VM) をデプロイできます。

注記

3 ノードクラスターをインストールする場合は、この手順をスキップしてください。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコントロールプレーンマシンで設定されます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
2. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

注記

vSphere インストール全体のすべての仮想マシン名が一意であることを確認してください。

3. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

4. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
5. **Select storage** タブで、設定ファイルとディスクファイル用のストレージを選択します。
6. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。
7. **Customize hardware** タブで、**Advanced Parameters** をクリックします。
   - **Attribute** フィールドおよび **Values** フィールドにデータを指定して、以下の設定パラメーター名と値を追加します。作成するパラメーターごとに **Add** ボタンを選択してください。
     - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
     - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
     - **disk.EnableUUID**: **TRUE** を指定します。
8. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。多くのネットワークが存在する場合は、**Add New Device** > **Network Adapter** を選択し、**New Network** メニュー項目に表示されるフィールドにネットワーク情報を入力します。
9. 残りの設定手順を完了します。**Finish** ボタンをクリックして、クローン作成操作を完了します。
10. **Virtual Machines** タブで仮想マシンを右クリックし、**Power** → **Power On** を選択します。

次のステップ

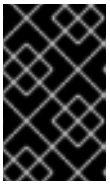
- 継続してクラスター用の追加のコンピュートマシンを作成します。

23.3.3.9. ディスクパーティション設定

ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして作成する場合にのみ正式にサポートされます。



重要

ディスクサイズが 100 GB を超える場合、特にディスクサイズが 1 TB を超える場合は、別の **/var** パーティションを作成します。詳細は、個別の **/var** パーティションの作成およびこの Red Hat ナレッジベースの記事 を参照してください。

重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

個別の /var パーティションの作成
一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
```

```
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、**manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

23.3.3.10. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
```

```
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

23.3.3.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

23.3.3.12. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

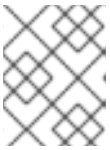
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

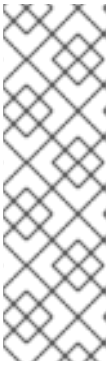
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

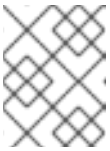
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

23.3.3.13. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                            4.16.0    True        False         False      40m
cluster-autoscaler                          4.16.0    True        False         False      37m
config-operator                             4.16.0    True        False         False      38m
```

```
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

23.3.3.13.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

23.3.3.13.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

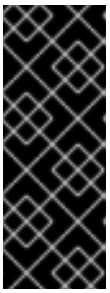
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

23.3.3.13.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

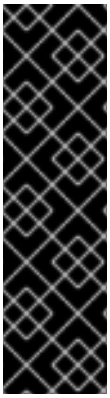
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Data Foundation など、クラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- "100Gi" の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリックストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

23.3.3.13.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

23.3.3.13.2.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、**1** つのレプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
```

```
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
4. 永続ボリューム要求 (PVC) のサイズ。

b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1. カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

正しい PVC を参照するようにレジストリーストレージを設定する手順は、vSphere のレジストリーの設定 を参照してください。

23.3.3.14. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace> 1
   ```

   1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

クラスターのインストールが完了したら、コンピュートマシンの vSphere への追加 に従って、コンピュートマシンをさらに追加できます。

23.3.3.15. コントロールプレーンノードの vSphere DRS 非アフィニティールールの設定

vSphere Distributed Resource Scheduler (DRS) 非アフィニティールールを設定して、OpenShift Container Platform コントロールプレーンノードでより高い可用性をサポートできます。非アフィニティールールにより、OpenShift Container Platform コントロールプレーンノードの vSphere 仮想マシンが同じ vSphere ノードにスケジュールされないようにします。

重要

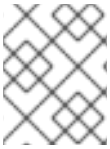
- 以下の情報はコンピュート DRS にのみ適用され、ストレージ DRS には適用されません。
- **govc** コマンドは、VMware で利用可能なオープンソースのコマンドであり、Red Hat からは利用できません。**govc** コマンドは、Red Hat サポートではサポートされません。
- **govc** のダウンロードおよびインストール手順は、VMware ドキュメントの Web サイト を参照してください。

以下のコマンドを実行して anti-affinity ルールを作成します。

コマンドの例

```
$ govc cluster.rule.create \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyCluster \
  -enable \
  -anti-affinity master-0 master-1 master-2
```

ルールを作成すると、コントロールプレーンノードは vSphere によって自動的に移行されるため、同じホストで実行されることはありません。vSphere が新しいルールを調整するまで、しばらく時間がかかる場合があります。コマンドを正しく補完する方法は、以下の手順に示します。



注記

移行は自動的に行われ、移行が完了するまで短い OpenShift API 停止またはレイテンシーが発生する可能性があります。

vSphere DRS の非アフィニティールールは、コントロールプレーンの仮想マシン名が変更された場合や、新しい vSphere クラスターへの移行時に手動で更新する必要があります。

手順

1. 以下のコマンドを実行して、既存の DRS 非アフィニティールールを削除します。

```
$ govc cluster.rule.remove \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyCluster
```

出力例

```
[13-10-22 09:33:24] Reconfigure /MyDatacenter/host/MyCluster...OK
```

2. 以下のコマンドを実行して、更新された名前でルールを再度作成します。

```
$ govc cluster.rule.create \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyOtherCluster \
  -enable \
  -anti-affinity master-0 master-1 master-2
```

23.3.3.16. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

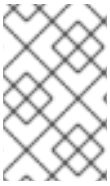
- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

23.3.3.17. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。
- オプション: 暗号化された仮想マシンを作成した場合は、暗号化されたストレージクラスを作成します。

23.3.4. ネットワークのカスタマイズによる vSphere へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、カスタマイズしたネットワーク設定オプションを使用して、独自にプロビジョニングする VMware vSphere インフラストラクチャーにクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。



注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

23.3.4.1. 前提条件

- user-provisioned infrastructure を使用したクラスターのインストールの準備 のタスクを完了した。
- VMware プラットフォームのライセンスを確認した。Red Hat は VMware ライセンスに制限を設けていませんが、一部の VMware インフラストラクチャーコンポーネントにはライセンスが必要です。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- インストールを完了するには、vSphere ホストに Red Hat Enterprise Linux CoreOS(RHCOS) OVA をアップロードする必要があります。このプロセスを完了するマシンには、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

23.3.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

23.3.4.3. VMware vSphere のリージョンとゾーンの有効化

OpenShift Container Platform クラスターを、単一の VMware vCenter で実行される複数の vSphere データセンターにデプロイできます。各データセンターは複数のクラスターを実行できます。この設定により、クラスターの障害を引き起こす可能性のあるハードウェア障害やネットワーク停止のリスクが軽減されます。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

重要

VMware vSphere のリージョンおよびゾーンの有効化機能には、クラスター内のデフォルトのストレージドライバーとして vSphere Container Storage Interface (CSI) ドライバーが必要です。そのため、この機能は新しくインストールされたクラスターでのみ使用できます。

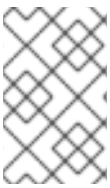
以前のリリースからアップグレードされたクラスターの場合は、クラスターの CSI 自動移行を有効にする必要があります。その後、アップグレードされたクラスターに対して複数のリージョンとゾーンを設定できます。

デフォルトのインストール設定では、クラスターが単一の vSphere データセンターにデプロイされます。クラスターを複数の vSphere データセンターにデプロイする場合は、リージョンおよびゾーン機能を有効にするインストール設定ファイルを作成する必要があります。

デフォルトの **install-config.yaml** ファイルには **vcenters フィールド** と **FailureDomains** フィールドが含まれており、OpenShift Container Platform クラスターに複数の vSphere データセンターとクラスターを指定できます。単一のデータセンターで設定される vSphere 環境に OpenShift Container Platform クラスターをインストールする場合は、これらのフィールドを空白のままにすることができます。

次のリストでは、クラスターのゾーンとリージョンの定義に関連する用語について説明します。

- 障害ドメイン: リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。
- リージョン: vCenter データセンターを指定します。リージョンを定義するには、**openshift-region** タグカテゴリーのタグを使用します。
- ゾーン: vCenter クラスターを指定します。ゾーンを定義するには、**openshift-zone** タグカテゴリーのタグを使用します。



注記

install-config.yaml ファイルで複数の障害ドメインを指定する予定がある場合は、設定ファイルを作成する前に、タグカテゴリー、ゾーンタグ、およびリージョンタグを作成する必要があります。

リージョンを表す vCenter データセンターごとに vCenter タグを作成する必要があります。さらに、データセンターで実行されるクラスターごとに、ゾーンを表す vCenter タグを作成する必要があります。タグを作成した後、各タグをそれぞれのデータセンターとクラスターにアタッチする必要があります。

次の表は、単一の VMware vCenter で実行されている複数の vSphere データセンターを含む設定のリージョン、ゾーン、タグ間の関係の例を示しています。

データセンター (リージョン)	クラスター (ゾーン)	タグ
米国東部	us-east-1	us-east-1a
		us-east-1b
	us-east-2	us-east-2a
		us-east-2b
us-west	us-west-1	us-west-1a
		us-west-1b
	us-west-2	us-west-2a
		us-west-2b

関連情報

- 追加の VMware vSphere 設定パラメーター
- 非推奨の VMware vSphere 設定パラメーター
- vSphere の自動移行
- VMware vSphere CSI ドライバー Operator

23.3.4.4. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

重要

Cloud Controller Manager Operator は、指定されたホスト名または IP アドレスに対して接続チェックを行います。到達可能な vCenter サーバーに対して、ホスト名または IP アドレスを指定していることを確認してください。存在しない vCenter サーバーにメタデータを提供すると、クラスターのインストールはブートストラップ段階で失敗します。

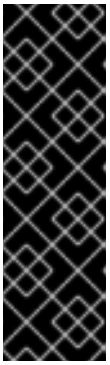
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

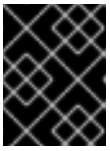
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- インストール設定パラメーター

23.3.4.4.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
additionalTrustBundlePolicy: Proxyonly
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- architecture: amd64
  name: <worker_node>
```

```
  platform: {}
  replicas: 0 3
controlPlane: 4
  architecture: amd64
  name: <parent_node>
  platform: {}
  replicas: 3 5
metadata:
  creationTimestamp: null
  name: test 6
networking:
---
platform:
  vsphere:
    failureDomains: 7
    - name: <failure_domain_name>
      region: <default_region_name>
      server: <fully_qualified_domain_name>
      topology:
        computeCluster: "/<datacenter>/host/<cluster>"
        datacenter: <datacenter> 8
        datastore: "/<datacenter>/datastore/<datastore>" 9
        networks:
        - <VM_Network_name>
        resourcePool: "/<datacenter>/host/<cluster>/Resources/<resourcePool>" 10
        folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 11
      zone: <default_zone_name>
    vcenters:
    - datacenters:
      - <datacenter>
      password: <password> 12
      port: 443
      server: <fully_qualified_domain_name> 13
      user: administrator@vsphere.local
    diskType: thin 14
fips: false 15
pullSecret: '{"auths": ...}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 4 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。両方のセクションで単一のマシンプールが定義されるため、使用されるコントロールプレーンは 1 つだけです。OpenShift Container Platform は、複数のコンピューティングプールの定義をサポートしていません。

3 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

5 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数

6 DNS レコードに指定したクラスター名。

7 リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。

8 vSphere データセンター。

9 仮想マシンファイル、テンプレート、ISO イメージを保持する vSphere データストアへのパス。

重要

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion はデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage vMotion を無効にする必要があります。

複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

10 オプション: インストーラーによってプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存のリソースプールの絶対パス (例: **/<datacenter_name>/host/<cluster_name>/Resources/<resource_pool_name>/<optional_nested_resource_pool_name>)**。値を指定しない場合、リソースはクラスターのルート **/example_datacenter/host/example_cluster/Resources** にインストールされます。

11 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供していて、**thin** という名前のデフォルトの **StorageClass** オブジェクトを使用したくない場合は、**install-config.yaml** ファイルから **folder** パラメーターを省略できます。

12 vSphere ユーザーに関連付けられたパスワード。

13 vCenter サーバーの完全修飾ホスト名または IP アドレス。

重要

Cloud Controller Manager Operator は、指定されたホスト名または IP アドレスに対して接続チェックを行います。到達可能な vCenter サーバーに対して、ホスト名または IP アドレスを指定していることを確認してください。存在しない vCenter サーバーにメタデータを提供すると、クラスターのインストールはブートストラップ段階で失敗します。

14 vSphere ディスクのプロビジョニング方法。

15

FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

16 OpenShift Cluster Manager から取得したプルシークレット。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

23.3.4.4.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

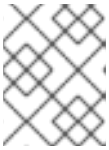
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
```

```
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

23.3.4.4.3. VMware vCenter のリージョンとゾーンの設定

デフォルトのインストール設定ファイルを変更して、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできるようにします。

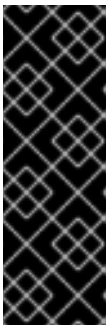
OpenShift Container Platform の以前のリリースのデフォルトの **install-config.yaml** ファイル設定は非推奨になりました。非推奨のデフォルト設定を引き続き使用できますが、**openshift-installer** により、設定ファイル内の非推奨のフィールドの使用を示す警告メッセージが表示されます。

重要

この例では、**govc** コマンドを使用します。**govc** コマンドは、VMware から入手できるオープンソースコマンドです。Red Hat からは入手できません。Red Hat サポートチームは **govc** コマンドを保守していません。**govc** のダウンロードとインストールの手順については、VMware ドキュメント Web サイトを参照してください。

前提条件

- 既存の **install-config.yaml** インストール設定ファイルがあります。

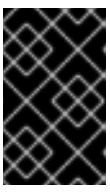


重要

VMware vCenter Server のデータセンターオブジェクトをプロビジョニングできるように、OpenShift Container Platform クラスターに少なくとも 1 つの障害ドメインを指定する必要があります。異なるデータセンター、クラスター、データストア、その他のコンポーネントに仮想マシンノードをプロビジョニングする必要がある場合は、複数の障害ドメインを指定することを検討してください。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

手順

1. 次の **govc** コマンドラインツールコマンドを入力して、**openshift-region** および **openshift-zone** vCenter タグカテゴリーを作成します。



重要

openshift-region および **openshift-zone** vCenter タグカテゴリーに異なる名前を指定すると、OpenShift Container Platform クラスターのインストールは失敗します。

```
$ govc tags.category.create -d "OpenShift region" openshift-region
```

```
$ govc tags.category.create -d "OpenShift zone" openshift-zone
```

2. クラスターをデプロイする各リージョン vSphere データセンターのリージョンタグを作成するには、ターミナルで次のコマンドを入力します。

```
$ govc tags.create -c <region_tag_category> <region_tag>
```

3. クラスターをデプロイする vSphere クラスターごとにゾーンタグを作成するには、次のコマンドを入力します。

```
$ govc tags.create -c <zone_tag_category> <zone_tag>
```

4. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにリージョンタグをアタッチします。

```
$ govc tags.attach -c <region_tag_category> <region_tag_1> /<datacenter_1>
```

5. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにゾーンタグをアタッチします。

```
$ govc tags.attach -c <zone_tag_category> <zone_tag_1> /<datacenter_1>/host/vcs-mdcnc-workload-1
```

6. インストールプログラムが含まれるディレクトリーに移動し、選択したインストール要件に従ってクラスターデプロイメントを初期化します。

vSphere センターで定義された複数のデータセンターを含むサンプル install-config.yaml ファイル

```
---
compute:
---
  vsphere:
     zones:
       - "<machine_pool_zone_1>"
       - "<machine_pool_zone_2>"
---
controlPlane:
---
vsphere:
     zones:
       - "<machine_pool_zone_1>"
       - "<machine_pool_zone_2>"
---
platform:
  vsphere:
    vcenters:
---
    datacenters:
      - <datacenter1_name>
      - <datacenter2_name>
    failureDomains:
    - name: <machine_pool_zone_1>
      region: <region_tag_1>
      zone: <zone_tag_1>
      server: <fully_qualified_domain_name>
```

```
      topology:
        datacenter: <datacenter1>
        computeCluster: "/<datacenter1>/host/<cluster1>"
        networks:
        - <VM_Network1_name>
        datastore: "/<datacenter1>/datastore/<datastore1>"
        resourcePool: "/<datacenter1>/host/<cluster1>/Resources/<resourcePool1>"
        folder: "/<datacenter1>/vm/<folder1>"
    - name: <machine_pool_zone_2>
      region: <region_tag_2>
      zone: <zone_tag_2>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter2>
        computeCluster: "/<datacenter2>/host/<cluster2>"
        networks:
        - <VM_Network2_name>
        datastore: "/<datacenter2>/datastore/<datastore2>"
        resourcePool: "/<datacenter2>/host/<cluster2>/Resources/<resourcePool2>"
        folder: "/<datacenter2>/vm/<folder2>"
---
```

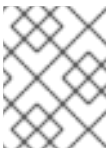
23.3.4.5. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

重要

CIDR 範囲 **172.17.0.0/16** は libVirt によって予約されています。この範囲、またはこの範囲と重複する範囲をクラスター内のネットワークに使用することはできません。

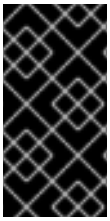
フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 でネットワークプラグインをさらにカスタマイズできます。

23.3.4.6. 高度なネットワーク設定の指定

ネットワークプラグインに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。



重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 次の例のように、**cluster-network-03-config.yml** ファイルでクラスターの高度なネットワーク設定を指定します。

OVN-Kubernetes ネットワークプロバイダーを有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
```

```
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

5. コントロールプレーンマシンおよび compute machineSets を定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- MachineSet ファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

23.3.4.7. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承します。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

クラスターネットワークプラグイン。**OVNKubernetes** は、インストール時にサポートされる唯一のプラグインです。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

23.3.4.7.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表23.29 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

フィールド	型	説明
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定

defaultNetwork オブジェクトの値は、以下の表で定義されます。

表23.30 **defaultNetwork** オブジェクト

フィールド	型	説明

フィールド	型	説明
type	**string**	**OVNKubernetes**。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。OpenShift SDN は、新しいクラスターのインストールの選択肢として利用できなくなりました。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドについて説明します。

表23.31 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。

フィールド	型	説明
ipsecConfig	**object**	IPsec 設定をカスタマイズするための設定オブジェクトを指定します。
ipv4	**object**	IPv4 設定の設定オブジェクトを指定します。
ipv6	**object**	IPv6 設定の設定オブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。  **注記** egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

表23.32 **ovnKubernetesConfig.ipv4** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **10** です。

フィールド	型	説明
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 デフォルト値は **100.64.0.0/16** です。

表23.33 **ovnKubernetesConfig.ipv6** object

フィールド	型	説明
internalTransitSwitchSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。east-west トラフィックを有効にする分散トランジットスイッチのサブネット。このサブネットは、OVN-Kubernetes またはホスト自体が使用する他のサブネットと重複できません。クラスター内のノードごとに 1 つの IP アドレスに対応するのに十分な大きさである必要があります。 デフォルト値は **fd98::/48** です。
internalJoinSubnet	string	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 デフォルト値は **fd98::/48** です。

表23.34 **policyAuditConfig オブジェクト**

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1 秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。
maxLogFiles	integer	保持されるログファイルの最大数。

フィールド	型	説明
比較先	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表23.35 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの egress トラフィックを送信するには、このフィールドを **true** に設定します。インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを**true**に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。
ipForwarding	**object**	**Network** リソースの **ipForwarding** 仕様を使用して、OVN-Kubernetes マネージドインターフェイス上のすべてのトラフィックの IP フォワーディングを制御できます。Kubernetes 関連のトラフィックの IP フォワーディングのみを許可するには、**Restricted** を指定します。すべての IP トラフィックの転送を許可するには、**Global** を指定します。新規インストールの場合、デフォルトは **Restricted** です。OpenShift Container Platform 4.14 以降に更新する場合、デフォルトは **Global** です。
ipv4	**object**	オプション：オブジェクトを指定して、IPv4 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

フィールド	型	説明
ipv6	**object**	オプション：オブジェクトを指定して、IPv6 アドレスのトラフィックにホストの内部 OVN-Kubernetes マスカレードアドレスを設定します。

表23.36 **gatewayConfig.ipv4** object

フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv4 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **10** です。

表23.37 **gatewayConfig.ipv6** object

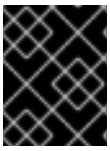
フィールド	型	説明
internalMasqueradeSubnet	**string**	ホストがトラフィックを処理できるようにするために内部で使用されるマスカレード IPv6 アドレス。ホストは、これらの IP アドレスと共有ゲートウェイブリッジインターフェイスで設定されます。デフォルト値は **fd69::/125** です。

表23.38 **ipsecConfig** オブジェクト

フィールド	型	説明
mode	**string**	IPsec 実装の動作を指定します。次の値のいずれかである必要があります。 • **Disabled**: クラスターノードで IPsec が有効になりません。 • **External**: 外部ホストとのネットワークトラフィックに対して IPsec が有効になります。 • **Full**: Pod トラフィックおよび外部ホストとのネットワークトラフィックに対して IPsec が有効になります。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
      ipsecConfig:
        mode: Full
```



重要

OVNKubernetes を使用すると、IBM Power® でスタック枯渇の問題が発生する可能性があります。

kubeProxyConfig オブジェクト設定 (OpenShiftSDN コンテナーネットワークインターフェイスのみ)
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表23.39 **kubeProxyConfig** オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

23.3.4.8. Ignition 設定ファイルの作成

クラスターマシンは手動で起動する必要があるため、クラスターがマシンを作成するために必要な Ignition 設定ファイルを生成する必要があります。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

- Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

install-config.yaml ファイルを作成している場合、それが含まれるディレクトリーを指定します。または、空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

23.3.4.9. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware vSphere でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

23.3.4.10. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。
- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。

2. ブートストラップノードの以下の二次的な Ignition 設定ファイルを、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。

- **<installation_directory>/master.ign**
- **<installation_directory>/worker.ign**
- **<installation_directory>/merge-bootstrap.ign**

4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

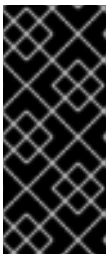
```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS OVA イメージを取得します。イメージは、RHCOS イメージミラー ページから入手できます。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。

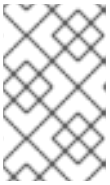
   a. **VMs and Templates** ビューをクリックします。

   b. データセンターの名前を右クリックします。

   c. **New Folder → New VM and Template Folder**をクリックします。

   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設定に適した場所にあるストレージを動的にプロビジョニングします。

7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。



注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。

   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。

   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。

   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

   e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

      - ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。

- **install-config.yaml** ファイルで指定したデータストアを選択します。
- 仮想マシンを暗号化する場合は、**Encrypt this virtual machine** を選択します。詳細については、「仮想マシンを暗号化するための要件」セクションを参照してください。

f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。

重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをコンピュートマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. 必要に応じて、仮想マシンテンプレートで設定された仮想ハードウェアバージョンを更新します。詳細は、VMware ドキュメントの Upgrading a virtual machine to the latest hardware version を参照してください。

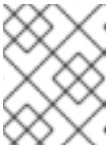
重要

必要に応じて、仮想マシンを作成する前に、仮想マシンテンプレートのハードウェアバージョンをバージョン 15 に更新することが推奨されます。vSphere で実行しているクラスターノード用にハードウェアバージョン 13 を使用することは非推奨となりました。インポートしたテンプレートがハードウェアバージョン 13 にデフォルト設定されている場合は、仮想マシンテンプレートをハードウェアバージョン 15 にアップグレードする前に、ESXi ホストが 6.7U3 以降を使用していることを確認する必要があります。vSphere のバージョンが 6.7U3 未満の場合は、このアップグレード手順を省略できます。ただし、OpenShift Container Platform の今後のバージョンでは、ハードウェアバージョン 13 および vSphere バージョンのサポートが 6.7U3 未満になる予定です。

9. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。



注記

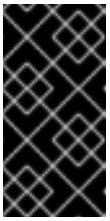
vSphere インストール全体のすべての仮想マシン名が一意であることを確認してください。

c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

e. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。

f. **Customize hardware** タブで、**Advanced Parameters** をクリックします。



重要

次の設定の提案は、例としてのみ使用されます。クラスター管理者は、クラスターに課せられるリソース需要に従ってリソースを設定する必要があります。クラスターリソースを最適に管理するには、クラスターのルートリソースプールからリソースプールを作成することを検討してください。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。
  - 静的 IP 設定を行います。

    コマンドの例

    ```
    $ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
    ```

    コマンドの例

    ```
    $ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none nameserver=8.8.8.8"
    ```

  - vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

    コマンドの例

    ```
    $ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-kargs=${IPCFG}"
    ```

- **Attribute** フィールドおよび **Values** フィールドにデータを指定して、以下の設定パラメーター名と値を追加します。作成するパラメーターごとに **Add** ボタンを選択してください。
  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。
  - クラスターの root リソースプールから子リソースプールを作成します。この子リソースプールでリソースの割り当てを実行します。

g. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更

g. Customize hardware タブで Virtual Hardware パネルを使用して、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

h. 残りの設定手順を完了します。**Finish** ボタンをクリックして、クローン作成操作を完了します。

i. **Virtual Machines** タブで仮想マシンを右クリックし、**Power → Power On** を選択します。

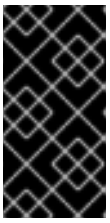
j. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

次のステップ

- 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

23.3.4.11. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

vSphere テンプレートを OpenShift Container Platform クラスターにデプロイした後に、そのクラスター内のマシンの仮想マシン (VM) をデプロイできます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
2. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

注記

vSphere インストール全体のすべての仮想マシン名が一意であることを確認してください。

3. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

4. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
5. **Select storage** タブで、設定ファイルとディスクファイル用のストレージを選択します。
6. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。
7. **Customize hardware** タブで、**Advanced Parameters** をクリックします。
   - **Attribute** フィールドおよび **Values** フィールドにデータを指定して、以下の設定パラメーター名と値を追加します。作成するパラメーターごとに **Add** ボタンを選択してください。
     - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
     - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
     - **disk.EnableUUID**: **TRUE** を指定します。
8. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。多くのネットワークが存在する場合は、**Add New Device** > **Network Adapter** を選択し、**New Network** メニュー項目に表示されるフィールドにネットワーク情報を入力します。
9. 残りの設定手順を完了します。**Finish** ボタンをクリックして、クローン作成操作を完了します。
10. **Virtual Machines** タブで仮想マシンを右クリックし、**Power** → **Power On** を選択します。

次のステップ

- 継続してクラスター用の追加のコンピュートマシンを作成します。

23.3.4.12. ディスクパーティション設定

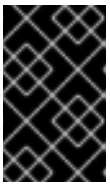
ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして作成する場合にのみ正式にサポートされます。

重要

ディスクサイズが 100 GB を超える場合、特にディスクサイズが 1 TB を超える場合は、別の **/var** パーティションを作成します。詳細は、個別の **/var** パーティションの作成およびこの Red Hat ナレッジベースの記事 を参照してください。



重要

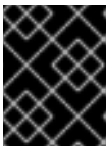
Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

個別の /var パーティションの作成
一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。



重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

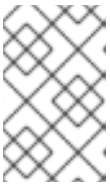
```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
```

```
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、**manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

23.3.4.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

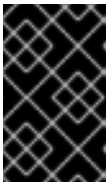
出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
```

```
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

23.3.4.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

23.3.4.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

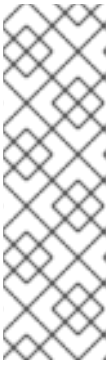
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

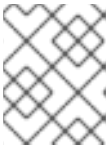
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

23.3.4.15.1. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
```

```
csi-snapshot-controller                 4.16.0    True        False         False      37m
dns                                     4.16.0    True        False         False      37m
etcd                                    4.16.0    True        False         False      36m
image-registry                          4.16.0    True        False         False      31m
ingress                                 4.16.0    True        False         False      30m
insights                                4.16.0    True        False         False      31m
kube-apiserver                          4.16.0    True        False         False      26m
kube-controller-manager                 4.16.0    True        False         False      36m
kube-scheduler                          4.16.0    True        False         False      36m
kube-storage-version-migrator           4.16.0    True        False         False      37m
machine-api                             4.16.0    True        False         False      29m
machine-approver                        4.16.0    True        False         False      37m
machine-config                          4.16.0    True        False         False      36m
marketplace                             4.16.0    True        False         False      37m
monitoring                              4.16.0    True        False         False      29m
network                                 4.16.0    True        False         False      38m
node-tuning                             4.16.0    True        False         False      37m
openshift-apiserver                     4.16.0    True        False         False      32m
openshift-controller-manager            4.16.0    True        False         False      30m
openshift-samples                       4.16.0    True        False         False      32m
operator-lifecycle-manager              4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog      4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver 4.16.0   True        False         False      32m
service-ca                              4.16.0    True        False         False      38m
storage                                 4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

23.3.4.15.2. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

23.3.4.15.3. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

23.3.4.15.3.1. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、**1** つのレプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
   2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
   3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
   4. 永続ボリューム要求 (PVC) のサイズ。

   b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

正しい PVC を参照するようにレジストリーストレージを設定する手順は、vSphere のレジストリーの設定 を参照してください。

23.3.4.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
```

```
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

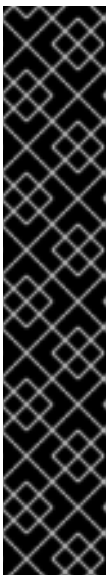
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                  NAME                                        READY   STATUS
```

```
RESTARTS   AGE
openshift-apiserver-operator      openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running   1          9m
openshift-apiserver               apiserver-67b9g                                 1/1     Running     0
3m
openshift-apiserver               apiserver-ljcmx                                 1/1     Running     0
1m
openshift-apiserver               apiserver-z25h4                                 1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

クラスターのインストールが完了したら、コンピュートマシンの vSphere への追加 に従って、コンピュートマシンをさらに追加できます。

23.3.4.17. コントロールプレーンノードの vSphere DRS 非アフィニティールールの設定

vSphere Distributed Resource Scheduler (DRS) 非アフィニティールールを設定して、OpenShift Container Platform コントロールプレーンノードでより高い可用性をサポートできます。非アフィニティールールにより、OpenShift Container Platform コントロールプレーンノードの vSphere 仮想マシンが同じ vSphere ノードにスケジュールされないようにします。

重要

- 以下の情報はコンピュート DRS にのみ適用され、ストレージ DRS には適用されません。
- **govc** コマンドは、VMware で利用可能なオープンソースのコマンドであり、Red Hat からは利用できません。**govc** コマンドは、Red Hat サポートではサポートされません。
- **govc** のダウンロードおよびインストール手順は、VMware ドキュメントの Web サイト を参照してください。

以下のコマンドを実行して anti-affinity ルールを作成します。

コマンドの例

```
$ govc cluster.rule.create \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyCluster \
  -enable \
  -anti-affinity master-0 master-1 master-2
```

ルールを作成すると、コントロールプレーンノードは vSphere によって自動的に移行されるため、同じホストで実行されることはありません。vSphere が新しいルールを調整するまで、しばらく時間がかかる場合があります。コマンドを正しく補完する方法は、以下の手順に示します。

注記

移行は自動的に行われ、移行が完了するまで短い OpenShift API 停止またはレイテンシーが発生する可能性があります。

vSphere DRS の非アフィニティールールは、コントロールプレーンの仮想マシン名が変更された場合や、新しい vSphere クラスターへの移行時に手動で更新する必要があります。

手順

1. 以下のコマンドを実行して、既存の DRS 非アフィニティールールを削除します。

```
$ govc cluster.rule.remove \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyCluster
```

出力例

```
[13-10-22 09:33:24] Reconfigure /MyDatacenter/host/MyCluster...OK
```

2. 以下のコマンドを実行して、更新された名前でルールを再度作成します。

```
$ govc cluster.rule.create \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyOtherCluster \
  -enable \
  -anti-affinity master-0 master-1 master-2
```

23.3.4.18. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

23.3.4.19. 次のステップ

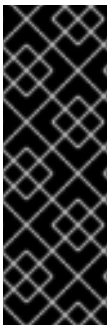
- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。
- オプション: 暗号化された仮想マシンを作成した場合は、暗号化されたストレージクラスを作成します。

23.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での vSphere へのクラスターのインストール

OpenShift Container Platform バージョン 4.16 では、制限されたネットワーク内で、独自にプロビジョニングする VMware vSphere インフラストラクチャーにクラスターをインストールできます。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

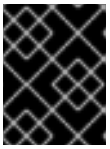


重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

23.3.5.1. 前提条件

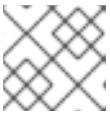
- user-provisioned infrastructure を使用したクラスターのインストールの準備 のタスクを完了した。
- VMware プラットフォームのライセンスを確認した。Red Hat は VMware ライセンスに制限を設けていませんが、一部の VMware インフラストラクチャーコンポーネントにはライセンスが必要です。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターの 永続ストレージ をプロビジョニングした。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- インストールを完了するには、vSphere ホストに Red Hat Enterprise Linux CoreOS(RHCOS) OVA をアップロードする必要があります。このプロセスを完了するマシンには、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。



注記

プロキシーを設定する場合は、このサイトリストも確認してください。

23.3.5.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.16 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェア、Nutanix、または VMware vSphere へのインストールに必要なインターネットアクセスが少なくて済む場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

23.3.5.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

23.3.5.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

23.3.5.4. VMware vSphere のリージョンとゾーンの有効化

OpenShift Container Platform クラスターを、単一の VMware vCenter で実行される複数の vSphere データセンターにデプロイできます。各データセンターは複数のクラスターを実行できます。この設定により、クラスターの障害を引き起こす可能性のあるハードウェア障害やネットワーク停止のリスクが軽減されます。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

重要

VMware vSphere のリージョンおよびゾーンの有効化機能には、クラスター内のデフォルトのストレージドライバーとして vSphere Container Storage Interface (CSI) ドライバーが必要です。そのため、この機能は新しくインストールされたクラスターでのみ使用できます。

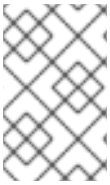
以前のリリースからアップグレードされたクラスターの場合は、クラスターの CSI 自動移行を有効にする必要があります。その後、アップグレードされたクラスターに対して複数のリージョンとゾーンを設定できます。

デフォルトのインストール設定では、クラスターが単一の vSphere データセンターにデプロイされます。クラスターを複数の vSphere データセンターにデプロイする場合は、リージョンおよびゾーン機能を有効にするインストール設定ファイルを作成する必要があります。

デフォルトの **install-config.yaml** ファイルには **vcenters フィールド** と **FailureDomains** フィールドが含まれており、OpenShift Container Platform クラスターに複数の vSphere データセンターとクラスターを指定できます。単一のデータセンターで設定される vSphere 環境に OpenShift Container Platform クラスターをインストールする場合は、これらのフィールドを空白のままにすることができます。

次のリストでは、クラスターのゾーンとリージョンの定義に関連する用語について説明します。

- 障害ドメイン: リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。
- リージョン: vCenter データセンターを指定します。リージョンを定義するには、**openshift-region** タグカテゴリーのタグを使用します。
- ゾーン: vCenter クラスターを指定します。ゾーンを定義するには、**openshift-zone** タグカテゴリーのタグを使用します。



注記

install-config.yaml ファイルで複数の障害ドメインを指定する予定がある場合は、設定ファイルを作成する前に、タグカテゴリー、ゾーンタグ、およびリージョンタグを作成する必要があります。

リージョンを表す vCenter データセンターごとに vCenter タグを作成する必要があります。さらに、データセンターで実行されるクラスターごとに、ゾーンを表す vCenter タグを作成する必要があります。タグを作成した後、各タグをそれぞれのデータセンターとクラスターにアタッチする必要があります。

次の表は、単一の VMware vCenter で実行されている複数の vSphere データセンターを含む設定のリージョン、ゾーン、タグ間の関係の例を示しています。

データセンター (リージョン)	クラスター (ゾーン)	タグ
米国東部	us-east-1	us-east-1a
		us-east-1b
	us-east-2	us-east-2a
		us-east-2b
us-west	us-west-1	us-west-1a
		us-west-1b
	us-west-2	us-west-2a
		us-west-2b

関連情報

- 追加の VMware vSphere 設定パラメーター
- 非推奨の VMware vSphere 設定パラメーター
- vSphere の自動移行
- VMware vSphere CSI ドライバー Operator

23.3.5.5. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

重要

Cloud Controller Manager Operator は、指定されたホスト名または IP アドレスに対して接続チェックを行います。到達可能な vCenter サーバーに対して、ホスト名または IP アドレスを指定していることを確認してください。存在しない vCenter サーバーにメタデータを提供すると、クラスターのインストールはブートストラップ段階で失敗します。

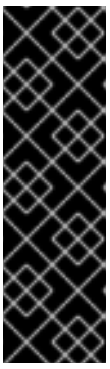
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。
- リポジトリーのミラーリングに使用するコマンドの出力で **imageContentSources** セクションを取得します。
- ミラーレジストリーの証明書の内容を取得する。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

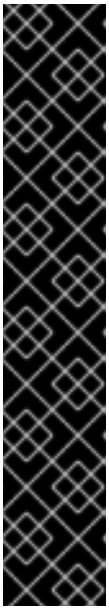
2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

- **docker.io** などの、RHCOS がデフォルトで信頼するレジストリーを使用しない限り、**additionalTrustBundle** セクションにミラーリポジトリーの証明書の内容を指定する必要があります。ほとんどの場合、ミラーの証明書を指定する必要があります。

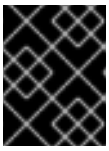
- リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを組み込む必要があります。



重要

- **ImageContentSourcePolicy** ファイルは、ミラーリングプロセスの終了後に **oc mirror** の出力として生成されます。
- **oc mirror** コマンドは、**ImageContentSourcePolicy** の定義に必要な YAML を含む **ImageContentSourcePolicy** ファイルを生成します。このファイルからテキストをコピーし、**install-config.yaml** ファイルに貼り付けます。
- 'oc mirror' コマンドを 2 回実行する必要があります。初めて **oc mirror** コマンドを実行すると、完全な **ImageContentSourcePolicy** ファイルが取得されます。**oc mirror** コマンドを 2 回目に実行すると、1 回目と 2 回目の実行の差のみが得られます。この動作のため、これらのファイルを 1 つの完全な **ImageContentSourcePolicy** ファイルにマージする必要がある場合に備えて、常にこれらのファイルのバックアップを保持する必要があります。これら 2 つの出力ファイルのバックアップを保持すると、完全な **ImageContentSourcePolicy** ファイルが確実に作成されます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

関連情報

- インストール設定パラメーター

23.3.5.5.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

```
additionalTrustBundlePolicy: Proxyonly
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- architecture: amd64
  name: <worker_node>
  platform: {}
  replicas: 0 (3)
controlPlane: (4)
  architecture: amd64
  name: <parent_node>
  platform: {}
  replicas: 3 (5)
metadata:
  creationTimestamp: null
  name: test (6)
```

```
networking:
---
platform:
  vsphere:
    failureDomains: 7
    - name: <failure_domain_name>
      region: <default_region_name>
      server: <fully_qualified_domain_name>
      topology:
        computeCluster: "/<datacenter>/host/<cluster>"
        datacenter: <datacenter> 8
        datastore: "/<datacenter>/datastore/<datastore>" 9
        networks:
        - <VM_Network_name>
        resourcePool: "/<datacenter>/host/<cluster>/Resources/<resourcePool>" 10
        folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 11
      zone: <default_zone_name>
    vcenters:
    - datacenters:
      - <datacenter>
      password: <password> 12
      port: 443
      server: <fully_qualified_domain_name> 13
      user: administrator@vsphere.local
    diskType: thin 14
fips: false 15
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
additionalTrustBundle: | 18
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 19
- mirrors:
  - <mirror_host_name>:<mirror_port>/<repo_name>/release
  source: <source_image_1>
- mirrors:
  - <mirror_host_name>:<mirror_port>/<repo_name>/release-images
  source: <source_image_2>
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 4 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。両方のセクションで単一のマシンプールが定義されるため、使用されるコントロールプレーンは 1 つだけです。OpenShift Container Platform は、複数のコンピューティングプールの定義をサポートしていません。

3 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロ

イする必要があります。

5 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

6 DNS レコードに指定したクラスター名。

7 リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。

8 vSphere データセンター。

9 仮想マシンファイル、テンプレート、ISO イメージを保持する vSphere データストアへのパス。

重要

データストアクラスター内に存在する任意のデータストアのパスを指定できます。デフォルトでは、Storage vMotion はデータストアクラスターに対して自動的に有効になります。Red Hat は Storage vMotion をサポートしていないため、OpenShift Container Platform クラスターのデータ損失の問題を回避するには、Storage vMotion を無効にする必要があります。

複数のデータストアにわたって仮想マシンを指定する必要がある場合は、**datastore** オブジェクトを使用して、クラスターの **install-config.yaml** 設定ファイルで障害ドメインを指定します。詳細は、「VMware vSphere のリージョンとゾーンの有効化」を参照してください。

10 オプション: インストーラーによってプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存のリソースプールの絶対パス (例: **/<datacenter_name>/host/<cluster_name>/Resources/<resource_pool_name>/<optional_nested_resource_pool_name>**)。値を指定しない場合、リソースはクラスターのルート **/example_datacenter/host/example_cluster/Resources** にインストールされます。

11 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供していて、**thin** という名前のデフォルトの **StorageClass** オブジェクトを使用したくない場合は、**install-config.yaml** ファイルから **folder** パラメーターを省略できます。

12 vSphere ユーザーに関連付けられたパスワード。

13 vCenter サーバーの完全修飾ホスト名または IP アドレス。

重要

Cloud Controller Manager Operator は、指定されたホスト名または IP アドレスに対して接続チェックを行います。到達可能な vCenter サーバーに対して、ホスト名または IP アドレスを指定していることを確認してください。存在しない vCenter サーバーにメタデータを提供すると、クラスターのインストールはブートストラップ段階で失敗します。

14 vSphere ディスクのプロビジョニング方法。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

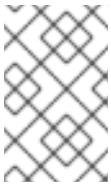
重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

16 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

18 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

19 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

23.3.5.5.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

23.3.5.5.3. VMware vCenter のリージョンとゾーンの設定

デフォルトのインストール設定ファイルを変更して、単一の VMware vCenter で実行される複数の vSphere データセンターに OpenShift Container Platform クラスターをデプロイできるようにします。

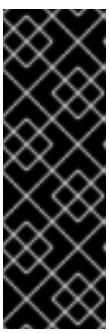
OpenShift Container Platform の以前のリリースのデフォルトの **install-config.yaml** ファイル設定は非推奨になりました。非推奨のデフォルト設定を引き続き使用できますが、**openshift-installer** により、設定ファイル内の非推奨のフィールドの使用を示す警告メッセージが表示されます。

重要

この例では、**govc** コマンドを使用します。**govc** コマンドは、VMware から入手できるオープンソースコマンドです。Red Hat からは入手できません。Red Hat サポートチームは **govc** コマンドを保守していません。**govc** のダウンロードとインストールの手順については、VMware ドキュメント Web サイトを参照してください。

前提条件

- 既存の **install-config.yaml** インストール設定ファイルがあります。

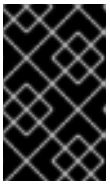


重要

VMware vCenter Server のデータセンターオブジェクトをプロビジョニングできるように、OpenShift Container Platform クラスターに少なくとも 1 つの障害ドメインを指定する必要があります。異なるデータセンター、クラスター、データストア、その他のコンポーネントに仮想マシンノードをプロビジョニングする必要がある場合は、複数の障害ドメインを指定することを検討してください。リージョンとゾーンを有効にするには、OpenShift Container Platform クラスターに複数の障害ドメインを定義する必要があります。

手順

1. 次の **govc** コマンドラインツールコマンドを入力して、**openshift-region** および **openshift-zone** vCenter タグカテゴリーを作成します。



重要

openshift-region および **openshift-zone** vCenter タグカテゴリーに異なる名前を指定すると、OpenShift Container Platform クラスターのインストールは失敗します。

```
$ govc tags.category.create -d "OpenShift region" openshift-region
```

```
$ govc tags.category.create -d "OpenShift zone" openshift-zone
```

2. クラスターをデプロイする各リージョン vSphere データセンターのリージョンタグを作成するには、ターミナルで次のコマンドを入力します。

```
$ govc tags.create -c <region_tag_category> <region_tag>
```

3. クラスターをデプロイする vSphere クラスターごとにゾーンタグを作成するには、次のコマンドを入力します。

```
$ govc tags.create -c <zone_tag_category> <zone_tag>
```

4. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにリージョンタグをアタッチします。

```
$ govc tags.attach -c <region_tag_category> <region_tag_1> /<datacenter_1>
```

5. 次のコマンドを入力して、各 vCenter データセンターオブジェクトにゾーンタグをアタッチします。

```
$ govc tags.attach -c <zone_tag_category> <zone_tag_1> /<datacenter_1>/host/vcs-mdcnc-workload-1
```

6. インストールプログラムが含まれるディレクトリーに移動し、選択したインストール要件に従ってクラスターデプロイメントを初期化します。

vSphere センターで定義された複数のデータセンターを含むサンプル install-config.yaml ファイル

```
---
compute:
---
  vsphere:
      zones:
        - "<machine_pool_zone_1>"
        - "<machine_pool_zone_2>"
---
controlPlane:
---
```

```
vsphere:
      zones:
        - "<machine_pool_zone_1>"
        - "<machine_pool_zone_2>"
---
platform:
  vsphere:
    vcenters:
---
    datacenters:
      - <datacenter1_name>
      - <datacenter2_name>
    failureDomains:
    - name: <machine_pool_zone_1>
      region: <region_tag_1>
      zone: <zone_tag_1>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter1>
        computeCluster: "/<datacenter1>/host/<cluster1>"
        networks:
        - <VM_Network1_name>
        datastore: "/<datacenter1>/datastore/<datastore1>"
        resourcePool: "/<datacenter1>/host/<cluster1>/Resources/<resourcePool1>"
        folder: "/<datacenter1>/vm/<folder1>"
    - name: <machine_pool_zone_2>
      region: <region_tag_2>
      zone: <zone_tag_2>
      server: <fully_qualified_domain_name>
      topology:
        datacenter: <datacenter2>
        computeCluster: "/<datacenter2>/host/<cluster2>"
        networks:
        - <VM_Network2_name>
        datastore: "/<datacenter2>/datastore/<datastore2>"
        resourcePool: "/<datacenter2>/host/<cluster2>/Resources/<resourcePool2>"
        folder: "/<datacenter2>/vm/<folder2>"
---
```

23.3.5.6. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシン、コンピュートマシンセット、およびコントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- コンピュートマシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、それが **false** に設定されていることを確認し

b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

23.3.5.7. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定する必要があります。

手順

1. **chrony.conf** ファイルのコンテンツを含む Butane 設定を作成します。たとえば、ワーカーノードで chrony を設定するには、**99-worker-chrony.bu** ファイルを作成します。

注記

Butane の詳細は、"Butane を使用したマシン設定の作成" を参照してください。

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-chrony 1
  labels:
    machineconfiguration.openshift.io/role: worker 2
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 3
    overwrite: true
    contents:
      inline: |
```

```
    pool 0.rhel.pool.ntp.org iburst 4
    driftfile /var/lib/chrony/drift
    makestep 1.0 3
    rtcsync
    logdir /var/log/chrony
```

1 2 コントロールプレーンノードでは、これらの両方の場所で **worker** の代わりに **master** を使用します。

3 マシン設定ファイルの **mode** フィールドに 8 進数の値でモードを指定します。ファイルを作成し、変更を適用すると、**mode** は 10 進数の値に変換されます。コマンド **oc get mc <mc-name> -o yaml** で YAML ファイルを確認できます。

4 DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-worker-chrony.yaml**) を生成します。

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 以下の 2 つの方法のいずれかで設定を適用します。

   - クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、**MachineConfig** オブジェクトファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を続行します。
   - クラスターがすでに実行中の場合は、ファイルを適用します。

```
$ oc apply -f ./99-worker-chrony.yaml
```

23.3.5.8. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware vSphere でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

23.3.5.9. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。
- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
2. ブートストラップノードの以下の二次的な Ignition 設定ファイルを、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

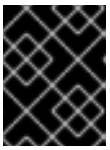
ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。
   - **<installation_directory>/master.ign**
   - **<installation_directory>/worker.ign**
   - **<installation_directory>/merge-bootstrap.ign**
4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
   たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS OVA イメージを取得します。イメージは、RHCOS イメージミラー ページから入手できます。

重要

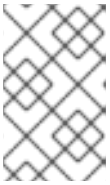
RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。

d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設定に適した場所にあるストレージを動的にプロビジョニングします。

7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。



注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。

b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。

c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。

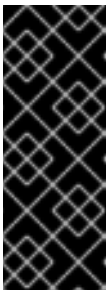
d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

- ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
- **install-config.yaml** ファイルで指定したデータストアを選択します。
- 仮想マシンを暗号化する場合は、**Encrypt this virtual machine** を選択します。詳細については、「仮想マシンを暗号化するための要件」セクションを参照してください。

f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

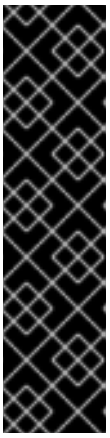
g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。



重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをコンピュートマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. 必要に応じて、仮想マシンテンプレートで設定された仮想ハードウェアバージョンを更新します。詳細は、VMware ドキュメントの Upgrading a virtual machine to the latest hardware version を参照してください。



重要

必要に応じて、仮想マシンを作成する前に、仮想マシンテンプレートのハードウェアバージョンをバージョン 15 に更新することが推奨されます。vSphere で実行しているクラスターノード用にハードウェアバージョン 13 を使用することは非推奨となりました。インポートしたテンプレートがハードウェアバージョン 13 にデフォルト設定されている場合は、仮想マシンテンプレートをハードウェアバージョン 15 にアップグレードする前に、ESXi ホストが 6.7U3 以降を使用していることを確認する必要があります。vSphere のバージョンが 6.7U3 未満の場合は、このアップグレード手順を省略できます。ただし、OpenShift Container Platform の今後のバージョンでは、ハードウェアバージョン 13 および vSphere バージョンのサポートが 6.7U3 未満になる予定です。

9. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

注記

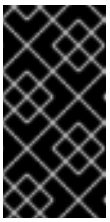
vSphere インストール全体のすべての仮想マシン名が一意であることを確認してください。

   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

   e. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。

   f. **Customize hardware** タブで、**Advanced Parameters** をクリックします。



重要

次の設定の提案は、例としてのみ使用されます。クラスター管理者は、クラスターに課せられるリソース需要に従ってリソースを設定する必要があります。クラスターリソースを最適に管理するには、クラスターのルートリソースプールからリソースプールを作成することを検討してください。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。
  - 静的 IP 設定を行います。

コマンドの例

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none
nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

コマンドの例

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

- vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

  コマンドの例

```
$ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-
kargs=${IPCFG}"
```

- **Attribute** フィールドおよび **Values** フィールドにデータを指定して、以下の設定パラメーター名と値を追加します。作成するパラメーターごとに **Add** ボタンを選択してください。
  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。
  - クラスターの root リソースプールから子リソースプールを作成します。この子リソースプールでリソースの割り当てを実行します。

g. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

h. 残りの設定手順を完了します。**Finish** ボタンをクリックして、クローン作成操作を完了します。

i. **Virtual Machines** タブで仮想マシンを右クリックし、**Power → Power On** を選択します。

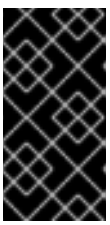
j. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

次のステップ

- 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

23.3.5.10. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

vSphere テンプレートを OpenShift Container Platform クラスターにデプロイした後に、そのクラスター内のマシンの仮想マシン (VM) をデプロイできます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
2. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

注記

vSphere インストール全体のすべての仮想マシン名が一意であることを確認してください。

3. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
4. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
5. **Select storage** タブで、設定ファイルとディスクファイル用のストレージを選択します。
6. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。
7. **Customize hardware** タブで、**Advanced Parameters** をクリックします。
   - **Attribute** フィールドおよび **Values** フィールドにデータを指定して、以下の設定パラメーター名と値を追加します。作成するパラメーターごとに **Add** ボタンを選択してください。
     - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
     - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
     - **disk.EnableUUID**: **TRUE** を指定します。
8. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。多くのネットワークが存在する場合は、**Add New Device** > **Network Adapter** を選択し、**New Network** メニュー項目に表示されるフィールドにネットワーク情報を入力します。
9. 残りの設定手順を完了します。**Finish** ボタンをクリックして、クローン作成操作を完了します。
10. **Virtual Machines** タブで仮想マシンを右クリックし、**Power → Power On** を選択します。

次のステップ

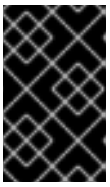
- 継続してクラスター用の追加のコンピュートマシンを作成します。

23.3.5.11. ディスクパーティション設定

ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

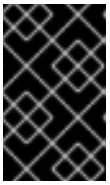
ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして作成する場合にのみ正式にサポートされます。



重要

ディスクサイズが 100 GB を超える場合、特にディスクサイズが 1 TB を超える場合は、別の **/var** パーティションを作成します。詳細は、個別の **/var** パーティションの作成およびこの Red Hat ナレッジベースの記事 を参照してください。



重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

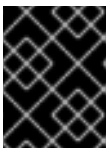
個別の /var パーティションの作成
一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。

- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。

- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。



重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

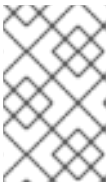
2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

23.3.5.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。

- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

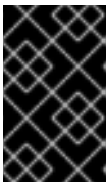
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

23.3.5.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

23.3.5.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME       AGE     REQUESTOR                                    CONDITION
```

```
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

23.3.5.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

23.3.5.15.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

23.3.5.15.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

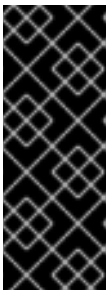
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

23.3.5.15.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Data Foundation など、クラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- "100Gi" の容量が必要です。

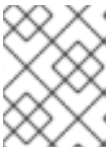
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリックストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります

す。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry    4.7                                 True        False         False      6h50m
```

23.3.5.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

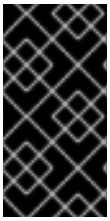
Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

23.3.5.15.2.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、**1** つのレプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
   2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
   3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
   4. 永続ボリューム要求 (PVC) のサイズ。

   b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

正しい PVC を参照するようにレジストリーストレージを設定する手順は、vSphere のレジストリーの設定 を参照してください。

23.3.5.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
```

```
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

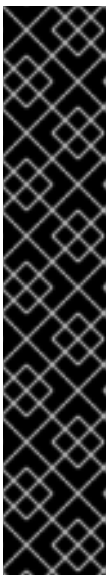
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                  NAME                                        READY   STATUS
```

```
RESTARTS   AGE
openshift-apiserver-operator      openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running   1          9m
openshift-apiserver               apiserver-67b9g                                 1/1     Running     0
3m
openshift-apiserver               apiserver-ljcmx                                 1/1     Running     0
1m
openshift-apiserver               apiserver-z25h4                                 1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

4. Cluster registration ページでクラスターを登録します。

クラスターのインストールが完了したら、コンピュートマシンの vSphere への追加 に従って、コンピュートマシンをさらに追加できます。

23.3.5.17. コントロールプレーンノードの vSphere DRS 非アフィニティールールの設定

vSphere Distributed Resource Scheduler (DRS) 非アフィニティールールを設定して、OpenShift Container Platform コントロールプレーンノードでより高い可用性をサポートできます。非アフィニティールールにより、OpenShift Container Platform コントロールプレーンノードの vSphere 仮想マシンが同じ vSphere ノードにスケジュールされないようにします。

重要

- 以下の情報はコンピュート DRS にのみ適用され、ストレージ DRS には適用されません。
- **govc** コマンドは、VMware で利用可能なオープンソースのコマンドであり、Red Hat からは利用できません。**govc** コマンドは、Red Hat サポートではサポートされません。
- **govc** のダウンロードおよびインストール手順は、VMware ドキュメントの Web サイト を参照してください。

以下のコマンドを実行して anti-affinity ルールを作成します。

コマンドの例

```
$ govc cluster.rule.create \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyCluster \
  -enable \
  -anti-affinity master-0 master-1 master-2
```

ルールを作成すると、コントロールプレーンノードは vSphere によって自動的に移行されるため、同じホストで実行されることはありません。vSphere が新しいルールを調整するまで、しばらく時間がかかる場合があります。コマンドを正しく補完する方法は、以下の手順に示します。

注記

移行は自動的に行われ、移行が完了するまで短い OpenShift API 停止またはレイテンシーが発生する可能性があります。

vSphere DRS の非アフィニティールールは、コントロールプレーンの仮想マシン名が変更された場合や、新しい vSphere クラスターへの移行時に手動で更新する必要があります。

手順

1. 以下のコマンドを実行して、既存の DRS 非アフィニティールールを削除します。

```
$ govc cluster.rule.remove \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyCluster
```

出力例

```
[13-10-22 09:33:24] Reconfigure /MyDatacenter/host/MyCluster...OK
```

2. 以下のコマンドを実行して、更新された名前でルールを再度作成します。

```
$ govc cluster.rule.create \
  -name openshift4-control-plane-group \
  -dc MyDatacenter -cluster MyOtherCluster \
  -enable \
  -anti-affinity master-0 master-1 master-2
```

23.3.5.18. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

23.3.5.19. 次のステップ

- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してクラスターに追加します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。
- オプション: 暗号化された仮想マシンを作成した場合は、暗号化されたストレージクラスを作成します。

23.4. ASSISTED INSTALLER を使用して VSPHERE にクラスターをインストールする

Assisted Installer を使用して、OpenShift Container Platform をオンプレミスのハードウェアまたはオンプレミスの VM にインストールできます。Assisted Installer を使用して OpenShift Container Platform をインストールすると、**x86_64**、**AArch64**、**ppc64le**、および **s390x** アーキテクチャーがサポートされます。

Assisted Installer は、Red Hat Hybrid Cloud Console で提供されるユーザーフレンドリーなインストールソリューションです。

23.4.1. 関連情報

- Assisted Installer を使用した OpenShift Container Platform のインストール

23.5. AGENT-BASED INSTALLER を使用して VSPHERE にクラスターをインストールする

エージェントベースのインストール方法では、選択した任意の方法でオンプレミスサーバーを柔軟に起動できます。Assisted Installation サービスの使いやすさと、エアギャップ環境を含むオフラインでの実行機能を兼ね備えています。

エージェントベースのインストールは、OpenShift Container Platform インストーラーのサブコマンドです。利用可能なリリースイメージを使用して、OpenShift Container Platform クラスターのデプロイに必要なすべての情報を含む起動可能な ISO イメージを生成します。

23.5.1. 関連情報

- Agent-based Installer を使用したインストールの準備

23.6. VSPHERE への 3 ノードクラスターのインストール

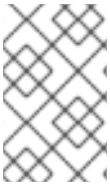
OpenShift Container Platform バージョン 4.16 では、VMware vSphere に 3 ノードクラスターをインストールできます。3 ノードクラスターは、コンピューティングマシンとしても機能する 3 つのコント

ロールプレーンマシンで設定されます。このタイプのクラスターは、クラスター管理者および開発者がテスト、開発、および実稼働に使用するためのより小さくリソース効率の高いクラスターを提供します。

インストーラーによってプロビジョニングされたインフラストラクチャーまたはユーザーによってプロビジョニングされたインフラストラクチャーのいずれかを使用して、3 ノードクラスターをインストールできます。

23.6.1. 3 ノードクラスターの設定

クラスターをデプロイする前に、**install-config.yaml** ファイルでワーカーノードの数を **0** に設定して、3 ノードクラスターを設定します。ワーカーノードの数を **0** に設定すると、コントロールプレーンマシンがスケジュール可能になります。これにより、アプリケーションワークロードをコントロールプレーンノードから実行するようにスケジュールできます。



注記

アプリケーションワークロードはコントロールプレーンノードから実行され、コントロールプレーンノードはコンピュートノードと見なされるため、追加のサブスクリプションが必要です。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

1. 次の **compute** スタンザに示すように、**install-config.yaml** ファイルでコンピューティングレプリカの数を **0** に設定します。

   3 ノードクラスターの install-config.yaml ファイルの例

   ```
   apiVersion: v1
   baseDomain: example.com
   compute:
   - name: worker
     platform: {}
     replicas: 0
   # ...
   ```

2. ユーザーがプロビジョニングしたインフラストラクチャーを使用して、クラスターをデプロイする場合:

   - HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするように、アプリケーションのイングレスロードバランサーを設定します。3 ノードクラスターでは、イングレスコントローラー Pod はコントロールプレーンノードで実行されます。詳細については、「ユーザーがプロビジョニングするインフラストラクチャーの負荷分散要件」を参照してください。

   - Kubernetes マニフェストファイルを作成したら、**cluster-scheduler-02-config.yml** ファイルで **spec.mastersSchedulable** パラメーターが **true** に設定されていることを確認します。このファイルは、**<installation_directory>/manifests** にあります。詳細については、「ユーザーがプロビジョニングしたインフラストラクチャーを使用した vSphere へのクラスターのインストール」の「Kubernetes マニフェストと Ignition 設定ファイルの作成」を参照してください。

- 追加のワーカーノードを作成しないでください。

3 ノードクラスターの cluster-scheduler-02-config.yml ファイルの例

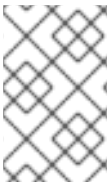
```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  creationTimestamp: null
  name: cluster
spec:
  mastersSchedulable: true
  policy:
    name: ""
status: {}
```

23.6.2. 次のステップ

- カスタマイズによる vSphere へのクラスターのインストール
- ユーザーによってプロビジョニングされるインフラストラクチャーを使用した vSphere へのクラスターのインストール

23.7. インストーラーでプロビジョニングされるインフラストラクチャーを使用した VSPHERE へのクラスターのインストール

インストーラーでプロビジョニングされるインフラストラクチャーを使用して、VMware vSphere インスタンスにデプロイしたクラスターを削除できます。



注記

openshift-install destroy cluster コマンドを実行して OpenShift Container Platform をアンインストールしても、vSphere ボリュームは自動的に削除されません。クラスター管理者は、vSphere ボリュームを手動で検索し、それらを削除する必要があります。

23.7.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

23.8. VSPHERE PROBLEM DETECTOR OPERATOR の使用

23.8.1. vSphere Problem Detector Operator について

vSphere Problem Detector Operator は、一般的なインストールおよびストレージに関連する正しくない設定の問題について vSphere にデプロイされたクラスターをチェックします。

Operator は **openshift-cluster-storage-operator** namespace で実行され、Cluster Storage Operator がクラスターが vSphere にデプロイされたことを検知すると Cluster Storage Operator によって起動します。vSphere Problem Detector Operator は vSphere vCenter Server と通信して、クラスター内の仮想マシン、デフォルトのデータストア、および vSphere vCenter Server 設定についての他の情報を判別します。Operator は Cloud Credential Operator からの認証情報を使用して vSphere に接続します。

Operator は以下のスケジュールに基づいてチェックを実行します。

- チェックは 8 時間ごとに実行されます。
- チェックに失敗すると、Operator は 1 分、2 分、4 分、8 分などの間隔でチェックを再び実行します。Operator は、8 時間を最大の間隔とし、その範囲内で間隔を 2 倍にします。
- すべてのチェックに合格すると、スケジュールは 8 時間の間隔に戻ります。

Operator は、障害の発生後にチェックの頻度を増加させ、Operator が障害状態が修復された直後に正常な状態を報告できるようにします。Operator を手動で実行し、トラブルシューティングについての情報をすぐに確認できます。

23.8.2. vSphere Problem Detector Operator チェックの実行

vSphere Problem Detector Operator のチェックを実行するスケジュールを上書きし、チェックを即時に実行できます。

vSphere Problem Detector Operator は 8 時間ごとにチェックを自動的に実行します。ただし、Operator が起動すると、チェックがすぐに実行されます。Operator は、Cluster Storage Operator の起動時に Cluster Storage Operator によって起動し、クラスターが vSphere で実行されているかどうかを判別します。チェックをすぐに実行するには、vSphere Problem Detector Operator を **0** にスケーリングしてから、**1** に戻し、vSphere Problem Detector Operator が再起動できるようにします。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. Operator を **0** にスケーリングします。

```
$ oc scale deployment/vsphere-problem-detector-operator --replicas=0 \
    -n openshift-cluster-storage-operator
```

デプロイメントがすぐにゼロにスケーリングされない場合、以下のコマンドを実行して Pod の終了を待機します。

```
$ oc wait pods -l name=vsphere-problem-detector-operator \
    --for=delete --timeout=5m -n openshift-cluster-storage-operator
```

2. Operator を **1** にスケーリングします。

```
$ oc scale deployment/vsphere-problem-detector-operator --replicas=1 \
    -n openshift-cluster-storage-operator
```

3. 古いリーダーロックを削除し、Cluster Storage Operator の新規リーダー選択を加速します。

```
$ oc delete -n openshift-cluster-storage-operator \
    cm vsphere-problem-detector-lock
```

検証

- vSphere Problem Detector Operator によって生成されるイベントまたはログを表示します。イベントまたはログに最新のタイムスタンプがあることを確認します。

23.8.3. vSphere Problem Detector Operator からのイベントの表示

vSphere Problem Detector Operator が設定チェックを実行した後に、コマンドラインまたは OpenShift Container Platform Web コンソールから表示できるイベントを作成します。

手順

- コマンドラインを使用してイベントを表示するには、以下のコマンドを実行します。

```
$ oc get event -n openshift-cluster-storage-operator \
    --sort-by={.metadata.creationTimestamp}
```

出力例

```
16m     Normal   Started         pod/vsphere-problem-detector-operator-xxxxx      Started
container vsphere-problem-detector
16m     Normal   Created         pod/vsphere-problem-detector-operator-xxxxx      Created
container vsphere-problem-detector
16m     Normal   LeaderElection  configmap/vsphere-problem-detector-lock  vsphere-
problem-detector-operator-xxxxx became leader
```

- OpenShift Container Platform Web コンソールを使用してイベントを表示するには、**Home → Events** に移動し、**Project** メニューから **openshift-cluster-storage-operator** を選択します。

23.8.4. vSphere Problem Detector Operator からのログの表示

vSphere Problem Detector Operator が設定チェックを実行した後に、コマンドラインまたは OpenShift Container Platform Web コンソールから表示できるログレコードを作成します。

手順

- コマンドラインを使用してログを表示するには、以下のコマンドを実行します。

```
$ oc logs deployment/vsphere-problem-detector-operator \
    -n openshift-cluster-storage-operator
```

出力例

```
I0108 08:32:28.445696       1 operator.go:209] ClusterInfo passed
I0108 08:32:28.451029       1 datastore.go:57] CheckStorageClasses checked 1 storage
classes, 0 problems found
I0108 08:32:28.451047       1 operator.go:209] CheckStorageClasses passed
I0108 08:32:28.452160       1 operator.go:209] CheckDefaultDatastore passed
I0108 08:32:28.480648       1 operator.go:271] CheckNodeDiskUUID:<host_name> passed
I0108 08:32:28.480685       1 operator.go:271] CheckNodeProviderID:<host_name> passed
```

- OpenShift Container Platform Web コンソールで Operator ログを表示するには、以下の手順を実行します。

  a. **Workloads → Pods** に移動します。

  b. **Projects** メニューから **openshift-cluster-storage-operator** を選択します。from the

  c. **vsphere-problem-detector-operator** Pod のリンクをクリックします。

  d. **Pod details** ページの **Logs** タブをクリックしてログを表示します。

23.8.5. vSphere Problem Detector Operator によって実行される設定チェック

以下の表は、vSphere Problem Detector Operator が実行する設定チェックを特定します。一部のチェックでは、クラスターの設定を確認します。他のチェックは、クラスター内の各ノードの設定を確認します。

表23.40 クラスター設定チェック

名前	説明
CheckDefaultDatastore	vSphere 設定のデフォルトのデータストア名が動的プロビジョニングで使用できる程度の短い名前であることを確認します。 このチェックに失敗した場合は、以下が予想されます。 ● **systemd** は、**Failed to set up mount unit: Invalid argument** などのエラーのログをジャーナルに記録します。 ● **systemd** は、仮想マシンがシャットダウンされていないか、ノードから すべての Pod をドレイン (解放) せずに再起動されている場合はボリュームをアンマウントしません。 このチェックに失敗した場合は、デフォルトのデータストアのより短い名前で vSphere を再設定します。
CheckFolderPermissions	デフォルトのデータストアでボリュームをリスト表示するパーミッションを検証します。このパーミッションは、ボリュームの作成に必要です。Operator は、/ および **/kubevols** ディレクトリーをリスト表示してパーミッションを検証します。ルートディレクトリーが存在する必要があります。これは、チェックの実行時に **/kubevols ディレクトリー** が存在しない場合に許可されます。**/kubevols** ディレクトリーは、このディレクトリーが存在しない場合に、データストアが動的プロビジョニングで使用される際に作成されます。 このチェックに失敗した場合は、OpenShift Container Platform のインストール時に指定された vCenter アカウントに必要なパーミッションを確認します。
CheckStorageClasses	以下を確認してください。 ● このストレージクラスによってプロビジョニングされる各永続ボリュームへの完全修飾パスは 255 文字未満です。 ● ストレージクラスがストレージポリシーを使用する場合、ストレージクラスは 1 つのポリシーのみを使用し、そのポリシーを定義する必要があります。
CheckTaskPermissions	最新のタスクおよびデータストアをリスト表示するパーミッションを検証します。
ClusterInfo	vSphere vCenter からクラスターバージョンおよび UUID を収集します。

表23.41 ノード設定チェック

名前	説明
CheckNodeDiskUUID	すべての vSphere 仮想マシンが **disk.enableUUID=TRUE** で設定されていることを確認します。 このチェックに失敗した場合は、Red Hat ナレッジベースソリューションの How to check 'disk.EnableUUID' parameter from VM in vSphere を参照してください。

名前	説明
CheckNodeProviderID	すべてのノードが vSphere vCenter の **ProviderID** で設定されていることを確認します。以下のコマンドからの出力に各ノードのプロバイダー ID が含まれていない場合に、このチェックに失敗します。 $ oc get nodes -o custom-columns=NAME:.metadata.name,PROVIDER_ID:.spec.providerID,UUID:.status.nodeInfo.systemUUID このチェックに失敗した場合は、クラスター内の各ノードのプロバイダー ID の設定方法について、vSphere の製品ドキュメントを参照してください。
CollectNodeESXiVersion	ノードを実行する ESXi ホストのバージョンを報告します。
CollectNodeHWVersion	ノードの仮想マシンのハードウェアバージョンを報告します。

23.8.6. ストレージクラス設定チェックについて

vSphere ストレージを使用する永続ボリュームの名前は、データストア名とクラスター ID に関連します。

永続ボリュームが作成されると、**systemd** は永続ボリュームのマウントユニットを作成します。**systemd** プロセスには、永続ボリュームに使用される VDMK ファイルへの完全修飾パスの長さについて 255 文字の制限があります。

完全修飾パスは、**systemd** および vSphere の命名規則に基づいています。命名規則では、以下のパターンを使用します。

```
/var/lib/kubelet/plugins/kubernetes.io/vsphere-volume/mounts/[<datastore>] 00000000-0000-0000-0000-000000000000/<cluster_id>-dynamic-pvc-00000000-0000-0000-0000-000000000000.vmdk
```

- 命名規則では、255 文字制限の内の 205 文字が必要です。
- データストア名とクラスター ID はデプロイメントで判別されます。
- データストア名とクラスター ID は前述のパターンに代入されます。次に、パスは特殊文字をエスケープできるように **systemd-escape** コマンドで処理されます。たとえば、ハイフン文字ではエスケープ後に 4 文字を使用します。エスケープされた値は **\x2d** になります。
- **systemd-escape** で処理した後に、**systemd** が VDMK ファイルへの完全修飾パスにアクセスできるようにするには、パスの長さが 255 文字未満である必要があります。

23.8.7. vSphere Problem Detector Operator のメトリック

vSphere Problem Detectorーリングスタックで使用される以下のメトリクスを公開します。

表23.42 vSphere Problem Detector Operator によって公開されるメトリック

名前	説明
vsphere_cluster_check_total	vSphere Problem Detector Operator が実行したクラスターレベルのチェックの累積数です。この数には、成功と失敗の両方が含まれます。
vsphere_cluster_check_errors	vSphere Problem Detector Operator が実行したクラスターレベルのチェックの失敗したチェック数です。たとえば、値 **1** は 1 つのクラスターレベルのチェックが失敗したことを示します。
vsphere_esxi_version_total	特定のバージョンを持つ ESXi ホストの数ホストが複数のノードを実行する場合は、ホストが 1 回のみカウントされることに注意してください。
vsphere_node_check_total	vSphere Problem Detector Operator が実行したノードレベルのチェックの累積数です。この数には、成功と失敗の両方が含まれます。
vsphere_node_check_errors	vSphere Problem Detector Operator が実行したノードレベルのチェックの失敗したチェック数です。たとえば、値 **1** は 1 つのノードレベルのチェックが失敗したことを示します。
vsphere_node_hw_version_total	特定のハードウェアバージョンを持つ vSphere ノードの数。
vsphere_vcenter_info	vSphere vCenter サーバーに関する情報

23.8.8. 関連情報

- モニタリングの概要

23.9. VSPHERE のインストール設定パラメーター

OpenShift Container Platform クラスターを vSphere にデプロイする前に、クラスターとそれをホストするプラットフォームをカスタマイズするためのパラメーターを指定します。**install-config.yaml** ファイルを作成するときは、コマンドラインを使用して必要なパラメーターの値を指定します。その後、**install-config.yaml** ファイルを変更して、クラスターをさらにカスタマイズできます。

23.9.1. vSphere で使用可能なインストール設定パラメーター

次の表に、インストールプロセスの一部として設定できる必須、オプション、および vSphere 固有のインストール設定パラメーターを示します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

23.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表23.43 必須パラメーター

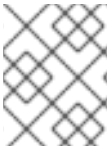
パラメーター	説明	値
apiVersion:	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	String
baseDomain:	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata:	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	Object
metadata: name:	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform:	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または {}。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	Object

パラメーター	説明	値
pullSecret:	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }

23.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

- Red Hat OpenShift Networking OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 と IPv6 の両方のアドレスファミリーがサポートされます。



注記

VMware vSphere では、デュアルスタックネットワークで IPv4 または IPv6 をプライマリーアドレスファミリーとして指定できます。

両方の IP アドレスファミリーを使用するようにクラスターを設定する場合は、次の要件を確認してください。

- どちらの IP ファミリーも、デフォルトゲートウェイに同じネットワークインターフェイスを使用する必要があります。
- 両方の IP ファミリーにデフォルトゲートウェイが必要です。
- すべてのネットワーク設定パラメーターに対して、IPv4 アドレスと IPv6 アドレスを同じ順序で指定する必要があります。たとえば、以下の設定では、IPv4 アドレスは IPv6 アドレスの前に記載されます。

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd00:10:128::/56
    hostPrefix: 64
  serviceNetwork:
  - 172.30.0.0/16
  - fd00:172:16::/112
```

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表23.44 ネットワークパラメーター

パラメーター	説明	値
networking:	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking: networkType:	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OVNKubernetes**。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。
networking: clusterNetwork:	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking: clusterNetwork: cidr:	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking: clusterNetwork: hostPrefix:	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking: serviceNetwork:	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OVN-Kubernetes ネットワークプラグインは、サービスネットワークに対して単一の IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking: machineNetwork:	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt と IBM Power® Virtual Server を除くすべてのプラットフォームのデフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。IBM Power® Virtual Server の場合、デフォルト値は **192.168.0.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

23.9.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

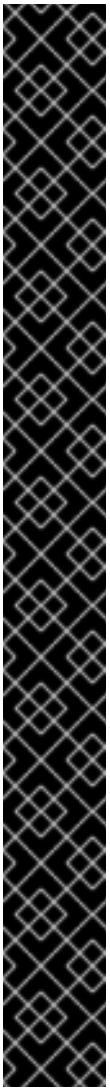
表23.45 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle:	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	String

パラメーター	説明	値
capabilities:	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳しくは、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities: baselineCapabilitySet:	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	String
capabilities: additionalEnabledCapabilities:	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
cpuPartitioningMode:	ワークロードパーティション設定を使用して、OpenShift Container Platform サービス、クラスター管理ワークロード、およびインフラストラクチャー Pod を分離し、予約された CPU セットで実行できます。ワークロードパーティショニングはインストール中にのみ有効にすることができ、インストール後に無効にすることはできません。このフィールドはワークロードのパーティショニングを有効にしますが、特定の CPU を使用するようにワークロードを設定するわけではありません。詳細は、**スケーラビリティとパフォーマンス** セクションの **ワークロードパーティショニング** ページを参照してください。	**None** または **AllNodes**。デフォルト値は **None** です。
compute:	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
compute: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
compute: name:	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute: platform:	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
compute: replicas:	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet:	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane:	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane: architecture:	プール内のマシンの命令セットアーキテクチャーを決定します。現在、さまざまなアーキテクチャーのクラスターはサポートされていません。すべてのプールは同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	String
controlPlane: name:	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**

パラメーター	説明	値
controlPlane: platform:	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**powervs**、**vsphere**、または **{}**
controlPlane: replicas:	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3**、シングルノード OpenShift をデプロイする場合は **1** です。
credentialsMode:	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。[1]

パラメーター	説明	値
fips:	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。 FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources:	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources: source:	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	String
imageContentSources: mirrors:	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish:	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。 **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。
sshKey:	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

1. すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**認証と認可** コンテンツの「クラウドプロバイダーの認証情報の管理」を参照してください。

23.9.1.4. 追加の VMware vSphere 設定パラメーター

追加の VMware vSphere 設定パラメーターは以下の表で説明されています。

表23.46 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform: vsphere:	クラスターをホストするクラウドプラットフォーム上のアカウントについて説明します。パラメーターを使用してプラットフォームをカスタマイズできます。マシンプール内のコンピュートマシンとコントロールプレーンマシンに追加の設定を指定する場合、このパラメーターは必要ありません。OpenShift Container Platform クラスターに指定できる vCenter サーバーは 1 つだけです。	vSphere 設定オブジェクトのディクショナリー
platform: vsphere: apiVIPs:	コントロールプレーン API アクセス用に設定した仮想 IP (VIP) アドレス。 **注記** このパラメーターは、installer-provisioned infrastructure にのみ適用されます。	複数の IP アドレス
platform: vsphere: diskType:	オプション: ディスクのプロビジョニング方法。この値が設定されていない場合、デフォルトで vSphere のデフォルトのストレージポリシーに設定されます。	有効な値は、**thin**、**thick**、または **eagerZeroedThick** です。
platform: vsphere: failureDomains:	リージョンとゾーン間の関係を確立します。障害ドメインは、**datastore** オブジェクトなどの vCenter オブジェクトを使用して定義します。障害ドメインは、OpenShift Container Platform クラスターノードの vCenter の場所を定義します。	障害ドメイン設定オブジェクトの配列。
platform: vsphere: failureDomains: name:	障害ドメインの名前。	String

パラメーター	説明	値
platform: vsphere: failureDomains: region:	クラスターに複数の障害ドメインを定義する場合は、タグを各 vCenter データセンターにアタッチする必要があります。リージョンを定義するには、**openshift-region** タグカテゴリーのタグを使用します。単一の vSphere データセンター環境の場合、タグをアタッチする必要はありませんが、パラメーターに英数字の値 (例: **datacenter**) を入力する必要があります。	String
platform: vsphere: failureDomains: server:	クライアントが障害ドメインリソースにアクセスできるように、VMware vCenter Server の完全修飾ホスト名または IP アドレスを指定します。**server** ロールを vSphere vCenter サーバーの場所に適用する必要があります。	String
platform: vsphere: failureDomains: zone:	クラスターに複数の障害ドメインを定義する場合は、各 vCenter クラスターにタグをアタッチする必要があります。ゾーンを定義するには、**openshift-zone** タグカテゴリーのタグを使用します。単一の vSphere データセンター環境の場合、タグをアタッチする必要はありませんが、パラメーターに英数字の値 (例: **cluster**) を入力する必要があります。	String
platform: vsphere: failureDomains: topology: computeCluster:	vSphere コンピュートクラスターへのパス。	String
platform: vsphere: failureDomains: topology: datacenter:	OpenShift Container Platform 仮想マシン (VM) が動作するデータセンターをリストして定義します。データセンターのリストは、**vcenters** フィールドで指定したデータセンターのリストと一致する必要があります。	String
platform: vsphere: failureDomains: topology: datastore:	障害ドメインの仮想マシンファイルを保存する vSphere データストアへのパスを指定します。**datastore** ロールを vSphere vCenter データストアの場所に適用する必要があります。	String

パラメーター	説明	値
platform: vsphere: failureDomains: topology: folder:	オプション: ユーザーが仮想マシンを作成する既存のフォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供していて、**thin** という名前のデフォルトの **StorageClass** オブジェクトを使用したくない場合は、**install-config.yaml** ファイルから **folder** パラメーターを省略できます。	String
platform: vsphere: failureDomains: topology: networks:	設定した仮想 IP アドレスと DNS レコードを含む vCenter インスタンス内のネットワークをリスト表示します。	String
platform: vsphere: failureDomains: topology: resourcePool:	オプション: このパラメーターは、インストールプログラムが仮想マシンを作成する既存のリソースプールの絶対パスを設定します (例: **/<datacenter_name>/host/<cluster_name>/Resources/<resource_pool_name>/<optional_nested_resource_pool_name>**)。値を指定しない場合、インストールプログラムは **/<datacenter_name>/host/<cluster_name>/Resources** の下のクラスターのルートにリソースをインストールします。	String
platform: vsphere: failureDomains: topology template:	既存の Red Hat Enterprise Linux CoreOS (RHCOS) イメージテンプレートまたは仮想マシンへの絶対パスを指定します。その後、インストールプログラムはイメージテンプレートまたは仮想マシンを使用して、vSphere ホストに RHCOS を迅速にインストールできます。RHCOS イメージを vSphere ホストにアップロードする代わりに、このパラメーターを使用することを検討してください。このパラメーターは、installer-provisioned infrastructure でのみ使用できます。	String
platform: vsphere: ingressVIPs:	クラスター Ingress 用に設定した仮想 IP (VIP) アドレス。 **注記** このパラメーターは、installer-provisioned infrastructure にのみ適用されます。	複数の IP アドレス

パラメーター	説明	値
platform: vsphere: vcenters:	サービスが vCenter サーバーと通信できるように接続の詳細を設定します。現在、単一の vCenter サーバーのみサポートされます。	vCenter 設定オブジェクトの配列。
platform: vsphere: vcenters: datacenters:	OpenShift Container Platform 仮想マシン (VM) が動作するデータセンターをリストして定義します。データセンターのリストは、**failureDomains** フィールドで指定されたデータセンターのリストと一致する必要があります。	String
platform: vsphere: vcenters: password:	vSphere ユーザーに関連付けられたパスワード。	String
platform: vsphere: vcenters: port:	vCenter サーバーとの通信に使用するポート番号。	Integer
platform: vsphere: vcenters: server:	vCenter サーバーの完全修飾ホスト名 (FQHN) または IP アドレス。	String
platform: vsphere: vcenters: user:	vSphere ユーザーに関連付けられたユーザー名。	String

23.9.1.5. 非推奨の VMware vSphere 設定パラメーター

OpenShift Container Platform 4.13 では、次の vSphere 設定パラメーターが非推奨になりました。これらのパラメーターは引き続き使用できますが、インストールプログラムはこれらのパラメーターを **install-config.yaml** ファイルに自動的に指定しません。

次の表に、非推奨になった各 vSphere 設定パラメーターを示します。

表23.47 非推奨の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform: vsphere: apiVIP:	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。 **注記** OpenShift Container Platform 4.12 以降では、**apiVIP** 設定は非推奨です。代わりに、**List** 形式を使用して、**apiVIPs** 設定に値を入力します。	IP アドレス (例: **128.0.0.1**)。
platform: vsphere: cluster:	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform: vsphere: datacenter:	OpenShift Container Platform 仮想マシン (VM) が動作するデータセンターを定義します。	文字列
platform: vsphere: defaultDatastore:	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列
platform: vsphere: folder:	オプション: インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられたフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform: vsphere: ingressVIP:	クラスター Ingress 用に設定した仮想 IP (VIP) アドレス。 **注記** OpenShift Container Platform 4.12 以降では、**ingressVIP** 設定は非推奨です。代わりに、**List** 形式を使用して、**ingressVIPs** 設定に値を入力します。	IP アドレス (例: **128.0.0.1**)。

パラメーター	説明	値
platform: vsphere: network:	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform: vsphere: password:	vCenter ユーザー名のパスワード。	文字列
platform: vsphere: resourcePool:	オプション: インストールプログラムが仮想マシンを作成する既存のリソースプールの絶対パス。値を指定しない場合、インストールプログラムは **/<datacenter_name>/host/<cluster_name>/Resources** の下のクラスターのルートにリソースをインストールします。	文字列 (例: **/<datacenter_name>/host/<cluster_name>/Resources/<resource_pool_name>/<optional_nested_resource_pool_name>)**。
platform: vsphere: username:	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform: vsphere: vCenter:	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列

23.9.1.6. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表23.48 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform: vsphere: clusterOSImage:	インストールプログラムが Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードする場所。このパラメーターのパス値を設定する前に、OpenShift Container Platform リリースのデフォルトの RHCOS ブートイメージが RHCOS イメージテンプレートまたは仮想マシンのバージョンと一致していることを確認してください。そうしないと、クラスターのインストールが失敗する可能性があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**

パラメーター	説明	値
platform: vsphere: osDisk: diskSizeGB:	ディスクのサイズ (ギガバイト単位)。	整数
platform: vsphere: cpus:	仮想マシンを割り当てる仮想プロセッサーコアの合計数**platform.vsphere.cpus** の値は、**platform.vsphere.coresPerSocket** 値の倍数である必要があります。	整数
platform: vsphere: coresPerSocket:	仮想マシンのソケットあたりのコア数。仮想マシンの仮想ソケットの数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。コントロールプレーンノードとワーカーノードのデフォルト値は、それぞれ **4** と **2** です。	整数
platform: vsphere: memoryMB:	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

第24章 任意のプラットフォームへのインストール

24.1. クラスターの任意のプラットフォームへのインストール

OpenShift Container Platform バージョン 4.16 では、仮想化およびクラウド環境を含め、独自にプロビジョニングする任意のインフラストラクチャーにクラスターをインストールできます。

重要

仮想化またはクラウド環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

24.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

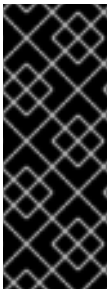
プロキシーを設定する場合は、このサイトリストも確認してください。

24.1.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.16 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにインストールパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

24.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

24.1.3.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表24.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピューティングマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

24.1.3.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表24.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピューティングマシンの使用は推奨されておらず、OpenShift Container Platform 4.10 以降では削除されています。

注記

OpenShift Container Platform バージョン 4.13 の時点で、RHCOS は RHEL バージョン 9.2 に基づいており、マイクロアーキテクチャーの要件を更新します。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、RHEL アーキテクチャー を参照してください。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

24.1.3.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管

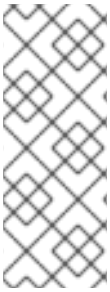
理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

24.1.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。



注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

24.1.3.4.1. DHCP を使用したクラスターノードのホスト名の設定

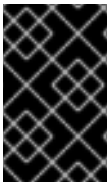
Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

24.1.3.4.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表24.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリック
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表24.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表24.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

24.1.3.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

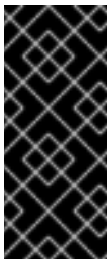
注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコード

で、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表24.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコードこれらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

24.1.3.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例24.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
```

```
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

   注記

   この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

-

例24.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

24.1.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

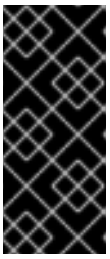
OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



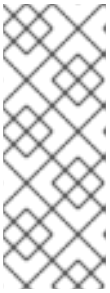
重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表24.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

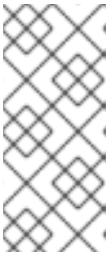
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表24.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック



注記

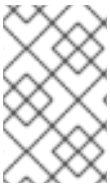
ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

24.1.3.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューショ

ンを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例24.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2 rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
```

```
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 5
    bind *:443
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:443 check inter 1s
    server compute1 compute1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 6
    bind *:80
    mode tcp
    balance source
    server compute0 compute0.ocp4.example.com:80 check inter 1s
    server compute1 compute1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

24.1.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用

の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。
   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

注記

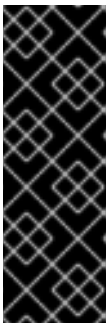
DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。



重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

24.1.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   注記

   出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

   random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

24.1.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** リストに追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

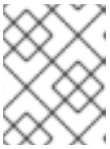
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

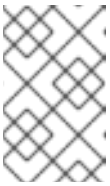
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

24.1.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使用してログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. インストールタイプのページに移動し、ホストオペレーティングシステムとアーキテクチャーに対応するインストールプログラムをダウンロードして、インストール設定ファイルを保存するディレクトリーにファイルを配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

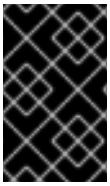
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

24.1.8. OpenShift CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

Linux への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

```
$ tar xvf <file>
```

6. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

Windows への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

macOC への OpenShift CLI のインストール
以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

24.1.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

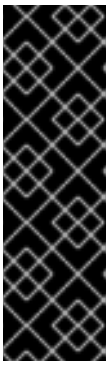
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

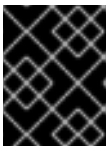
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

24.1.9.1. 他のプラットフォーム用のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、必要なパラメーターの値を変更することができます。

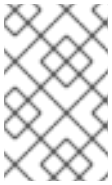
```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OVNKubernetes (11)
```

```
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths": ...}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



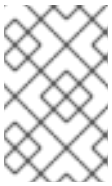
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネット

ワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

注記

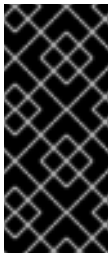
クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。プラットフォーム用に追加のプラットフォーム設定変数を指定することはできません。



重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

15 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

24.1.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

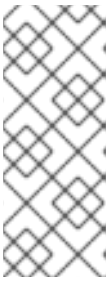
1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション： **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。



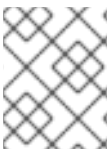
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

24.1.9.3. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで設定されるベアメタルクラスターに、ゼロコンピュートマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

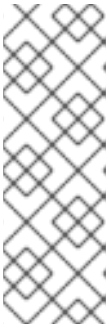
前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```



注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

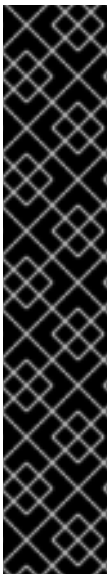
- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

24.1.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

24.1.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

RHCOS をマシンにインストールするには、ISO イメージまたはネットワーク PXE ブートを使用する手順のいずれかを実行します。

注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL8 コンピュートマシンのみがサポートされています。

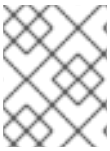
以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブインストール起動プロセスを中断してカーネル引数を追加できます。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。

- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールするノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。この特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。



注記

OpenShift Container Platform 4.6 の時点で、RHCOS ISO およびその他のインストールアーティファクトは、4K セクターのディスクへのインストールをサポートします。

24.1.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

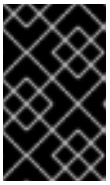
手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

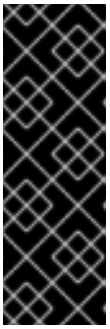
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



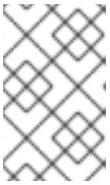
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。



注記

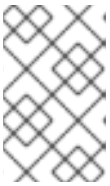
RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、**sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。



注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化しま

以下の例では、**/dev/sda** ディスクへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

重要

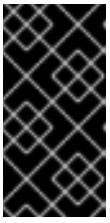
OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後、システムを再起動する必要があります。システムの再起動後、指定した Ignition 設定ファイルを適用します。
10. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

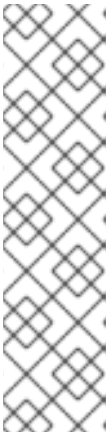
11. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

24.1.11.2. PXE または iPXE ブートを使用した RHCOS のインストール

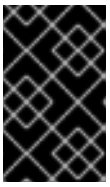
PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE または iPXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

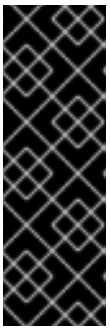
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+
(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.16/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs**、**kernel**、および **initramfs** ファイルを HTTP サーバーにアップロードします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE または iPXE インストールを設定し、インストールを開始します。ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

- PXE(**x86_64**) の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

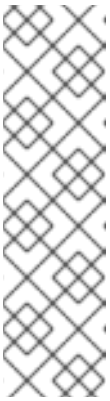
- iPXE (**x86_64** + **aarch64**) の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

1. HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

3. HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

注記

aarch64 アーキテクチャーで CoreOS **kernel** をネットワークブートするには、**IMAGE_GZIP** オプションが有効になっているバージョンの iPXE ビルドを使用する必要があります。iPXE の **IMAGE_GZIP** オプション を参照してください。

- **aarch64** 上の PXE (第 2 段階として UEFI と Grub を使用) の場合:

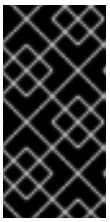
```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
    initrd rhcos-<version>-live-initramfs.<architecture>.img 3
}
```

1. HTTP/TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は、TFTP サーバー上の **kernel** ファイルの場所になります。**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値は HTTP サーバー上のブートストラップ Ignition 設定ファイルの場所になります。

2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

3. TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

7. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

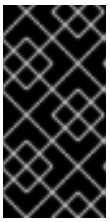
8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

9. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

10. クラスターのマシンの作成を続行します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

24.1.11.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。本セクションでは、以下のような手法で実行できるいくつかの設定について説明します。

- カーネル引数をライブインストーラーに渡す

- ライブシステムからの **coreos-installer** の手動による実行
- ライブ ISO または PXE ブートイメージのカスタマイズ

本セクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

24.1.11.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

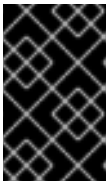
- 詳細の RHCOS インストールリファレンスの表を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

手順

1. ISO インストーラーを起動します。
2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。
3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```



重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

4. インストール済みのシステムで再起動します。

関連情報

- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

24.1.11.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。デフォルトのパーティション設定をオーバーライドしない限り、特定のアーキテクチャーの各 RHCOS ノードで同じパーティションレイアウトが使用されます。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。

重要

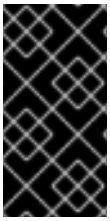
ノードでカスタムパーティションスキームを使用すると、OpenShift Container Platform が一部のノードパーティションでモニタリングやアラートを行わなくなる可能性があります。デフォルトのパーティション設定をオーバーライドする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

OpenShift Container Platform は、次の 2 つのファイルシステム識別子を監視します。

- **nodefs**: **/var/lib/kubelet** を含むファイルシステム
- **imagefs**: **/var/lib/containers** を含むファイルシステム

デフォルトのパーティションスキームの場合、**nodefs** と **imagefs** は同じルートファイルシステム (/) を監視します。

RHCOS を OpenShift Container Platform クラスターノードにインストールするときにデフォルトのパーティション設定をオーバーライドするには、別のパーティションを作成する必要があります。コンテナーとコンテナーイメージ用に別のストレージパーティションを追加する状況を考えてみましょう。たとえば、**/var/lib/containers** を別のパーティションにマウントすると、kubelet が **/var/lib/containers** を **imagefs** ディレクトリーとして、ルートファイルシステムを **nodefs** ディレクトリーとして個別に監視します。



重要

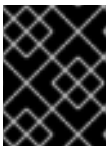
より大きなファイルシステムをホストするためにディスクサイズを変更した場合は、別の **/var/lib/containers** パーティションを作成することを検討してください。多数の割り当てグループによって発生する CPU 時間の問題を軽減するには、**xfs** 形式のディスクのサイズを変更することを検討してください。

24.1.11.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。



重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

1. インストールホストで、OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定が

ない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の /**var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

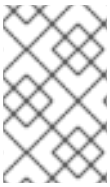
次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

24.1.11.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、**coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。



注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/disk/by-id/scsi-<serial_number>
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/disk/by-id/scsi-<serial_number>
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/disk/by-id/scsi-<serial_number>
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

24.1.11.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。

  重要

  これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

  PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

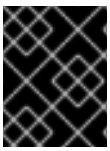
- **Live install Ignition config**: このタイプは、**coreos-installer customize** サブコマンドとそのさまざまなオプションを使用して作成できます。この方法では、Ignition 設定はライブインストールメディアに渡され、起動直後に実行され、RHCOS システムがディスクにインストールされる前または後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。
  PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、 Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

24.1.11.3.4. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

24.1.11.3.4.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。



重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

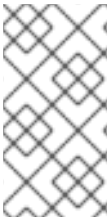
ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1つまたは複数のネットワークを設定する場合、1つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

  ```
  ip=::10.10.10.254::::
  ```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

  ```
  rd.route=20.20.20.0/24:20.20.20.254:enp2s0
  ```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

  ```
  ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
  vlan=enp2s0.100:enp2s0
  ```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、**options** はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

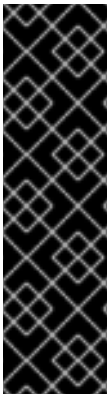
  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合する



重要

SR-IOV デバイスの NIC パーティショニングの有効化に関連する Day 1 操作のサポートは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

オプション: **bond=** オプションを使用して、複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合できます。

各ノードで、次のタスクを実行する必要があります。

1. SR-IOV デバイスの管理 のガイダンスに従って、SR-IOV 仮想機能 (VF) を作成します。「仮想マシンへの SR-IOV ネットワークデバイスの接続」セクションの手順に従います。

2. ボンドを作成し、目的の VF をボンドに接続し、ネットワークボンディングの設定 のガイダンスに従って、ボンドリンクの状態を設定します。説明されている手順のいずれかに従って、結合を作成します。

次の例は、使用する必要がある構文を示しています。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
<name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は仮想機能 (VF) をカーネル内の既知の名前で表し、**ip link** コマンド (**eno1f0**、**eno2f0**) の出力に表示されます。**options** は結合オプションのコンマ区切りリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

  - DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=bond0:dhcp
    ```

  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
name はチームデバイス名 (**team0**)、network_interfacesは物理 (イーサネット) インターフェイス (**em1**、**em2**) のコンマ区切りリストを表します。

注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

24.1.11.3.4.2. ISO および PXE インストール用のcoreos-installer オプション

RHCOS は、ISO イメージから RHCOS ライブ環境に起動した後に、コマンドプロンプトで **coreos-installer install <options> <device>** を実行してインストールできます。

以下の表は、**coreos-installer** コマンドに渡すことのできるサブコマンド、オプションおよび引数を示しています。

表24.9 coreos-installer サブコマンド、コマンドラインオプション、および引数

coreos-installer install サブコマンド	
サブコマンド	**説明**
$ coreos-installer install <options> <device>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer install サブコマンドオプション	
オプション	**説明**
-u、--image-url <url>	イメージの URL を手動で指定します。
-f、--image-file <path>	ローカルイメージファイルを手動で指定します。デバッグに使用されます。
-i、--ignition-file <path>	ファイルから Ignition 設定を埋め込みます。
-l、--ignition-url <URL>	URL から Ignition 設定を埋め込みます。
--ignition-hash <digest>	Ignition 設定の **type-value** をダイジェスト値を取得します。
-p、--platform <name>	インストール済みシステムの Ignition プラットフォーム ID を上書きします。
--console <spec>	インストールされたシステムのカーネルとブートローダーコンソールを設定します。**<spec>** の形式の詳細については、Linux カーネルシリアルコンソール のドキュメントを参照してください。
--append-karg <arg>...	インストール済みシステムにデフォルトのカーネル引数を追加します。
--delete-karg <arg>...	インストール済みシステムからデフォルトのカーネル引数を削除します。

-n、--copy-network	インストール環境からネットワーク設定をコピーします。  **重要** **--copy-network** オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。
--network-dir <path>	**-n** を指定して使用する場合。デフォルトは **/etc/NetworkManager/system-connections/** です。
--save-partlabel <lx>..	このラベル glob でパーティションを保存します。
--save-partindex <id>…	この数または範囲でパーティションを保存します。
--insecure	RHCOS イメージ署名の検証を省略します。
--insecure-ignition	HTTPS またはハッシュなしで Ignition URL を許可します。
--architecture <name>	ターゲット CPU アーキテクチャー。有効な値は **x86_64** および **aarch64** です。
--preserve-on-error	エラー時のパーティションテーブルは消去しないでください。
-h、--help	ヘルプ情報を表示します。
coreos-installer **インストールサブコマンド引数**	
引数	**説明**
<device>	宛先デバイス。
coreos-installer **ISO サブコマンド**	
サブコマンド	**説明**
$ coreos-installer iso customize <options> <ISO_image>	RHCOS ライブ ISO イメージをカスタマイズします。
coreos-installer iso reset <options> <ISO_image>	RHCOS ライブ ISO イメージをデフォルト設定に復元します。

coreos-installer iso ignition remove <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を削除します。
coreos-installer ISO カスタマイズサブコマンドオプション	
オプション	**説明**
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。
--dest-karg-append <arg>	宛先システムの各起動にカーネル引数を追加します。
--dest-karg-delete <arg>	宛先システムの各起動からカーネル引数を削除します。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。
--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
--post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
--live-karg-append <arg>	ライブ環境の各ブートにカーネル引数を追加します。
--live-karg-delete <arg>	ライブ環境の各ブートからカーネル引数を削除します。

--live-karg-replace <k=o=n>	ライブ環境の各起動で、**key=old=new** の形式でカーネル引数を置き換えます。
-f, --force	既存の Ignition 設定を上書きします。
-o、--output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。
coreos-installer PXE サブコマンド	
サブコマンド	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
coreos-installer pxe customize <options> <path>	RHCOS ライブ PXE ブート設定をカスタマイズします。
coreos-installer pxe ignition wrap <options>	イメージに Ignition 設定をラップします。
coreos-installer pxe ignition unwrap <options> <image_name>	イメージでラップされた Ignition 設定を表示します。
coreos-installer PXE はサブコマンドオプションをカスタマイズします	
オプション	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。
--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。

post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
-o, --output <path>	initramfs を新しい出力ファイルに書き込みます。  **注記** このオプションは、PXE 環境に必要です。
-h、--help	ヘルプ情報を表示します。

24.1.11.3.4.3. ISO または PXE インストールのcoreos.inst ブートオプション

coreos.inst ブートパラメーターを RHCOS ライブインストーラーに渡して、ブート時に **coreos-installer** オプションを自動的に起動できます。これらは、標準のブート引数の追加として提供されます。

- ISO インストールの場合、ブートローダーメニューで自動ブートを中断して **coreos.inst** オプションを追加できます。**RHEL CoreOS (Live)** メニューオプションが強調表示されている状態で **TAB** を押すと、自動ブートを中断できます。
- PXE または iPXE インストールの場合、RHCOS ライブインストーラーのブート前に **coreos.inst** オプションを **APPEND** 行に追加する必要があります。

以下の表は、ISO および PXE インストールの RHCOS ライブインストーラーの **coreos.inst** ブートオプションを示しています。

表24.10 coreos.inst ブートオプション

引数	説明
coreos.inst.install_dev	必須。インストール先のシステムのブロックデバイス。**sda** は許可されていますが、**/dev/sda** などの完全パスを使用することが推奨されます。
coreos.inst.ignition_url	オプション: インストール済みシステムに埋め込む Ignition 設定の URL。URL が指定されていない場合、Ignition 設定は埋め込まれません。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

引数	説明
coreos.inst.save_partlabel	オプション: インストール時に保存するパーティションのコンマ区切りのラベル。glob 形式のワイルドカードが許可されます。指定したパーティションは存在する必要はありません。
coreos.inst.save_partindex	オプション: インストール時に保存するパーティションのコンマ区切りのインデックス。範囲 **m-n** は許可され、**m** または **n** のいずれかを省略できます。指定したパーティションは存在する必要はありません。
coreos.inst.insecure	オプション: **coreos.inst.image_url** で署名なしと指定される OS イメージを許可します。
coreos.inst.image_url	オプション: 指定した RHCOS イメージをダウンロードし、インストールします。 ● この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。 ● この引数は、ライブメディアに一致しないバージョンの RHCOS をインストールするために使用できますが、インストールするバージョンに一致するメディアを使用することが推奨されます。 ● **coreos.inst.image_url** を使用している場合は、**coreos.inst.insecure** も使用する必要があります。これは、ベアメタルメディアが OpenShift Container Platform について GPG で署名されていないためです。 ● HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.skip_reboot	オプション: システムはインストール後に再起動しません。インストールが完了するとプロンプトが表示され、インストール時に生じる内容を検査できます。この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。
coreos.inst.platform_id	オプション: RHCOS イメージがインストールされるプラットフォームの Ignition プラットフォーム ID。デフォルトは **metal** です。このオプションは、VMware などのクラウドプロバイダーから Ignition 設定を要求するかどうかを決定します。例: **coreos.inst.platform_id=vmware**

引数	説明
ignition.config.url	オプション: ライブ起動の Ignition 設定の URL。たとえば、これは **coreos-installer** の起動方法をカスタマイズしたり、インストール前後にコードを実行するために使用できます。これはインストール済みシステムの Ignition 設定である **coreos.inst.ignition_url** とは異なります。

24.1.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.29.4 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

24.1.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

24.1.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

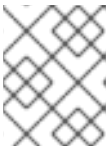
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

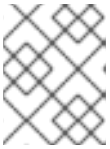
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

24.1.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                            4.16.0    True        False         False      40m
cluster-autoscaler                          4.16.0    True        False         False      37m
config-operator                             4.16.0    True        False         False      38m
```

```
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

24.1.15.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

24.1.15.2. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

24.1.15.3. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

24.1.15.3.1. ベアメタルおよび他の手動インストールの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- ベアメタルなどの、手動でプロビジョニングされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードを使用するクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

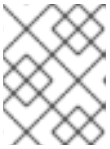
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION           AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.16              True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

24.1.15.3.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

24.1.15.3.3. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

> **重要**
>
> ブロックストレージボリューム (または永続ボリューム) はサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。
>
> イメージレジストリーでブロックストレージボリュームを使用することを選択した場合は、ファイルシステムの persistent volume claim (PVC) を使用する必要があります。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、1つの (**1**) レプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

   (3) 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (4) 永続ボリューム要求 (PVC) のサイズ。

   b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: (1)
```

(1) カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

24.1.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.16.0    True        False         False      19m
baremetal                                  4.16.0    True        False         False      37m
cloud-credential                           4.16.0    True        False         False      40m
cluster-autoscaler                         4.16.0    True        False         False      37m
config-operator                            4.16.0    True        False         False      38m
console                                    4.16.0    True        False         False      26m
csi-snapshot-controller                    4.16.0    True        False         False      37m
dns                                        4.16.0    True        False         False      37m
etcd                                       4.16.0    True        False         False      36m
image-registry                             4.16.0    True        False         False      31m
ingress                                    4.16.0    True        False         False      30m
insights                                   4.16.0    True        False         False      31m
kube-apiserver                             4.16.0    True        False         False      26m
kube-controller-manager                    4.16.0    True        False         False      36m
kube-scheduler                             4.16.0    True        False         False      36m
kube-storage-version-migrator              4.16.0    True        False         False      37m
machine-api                                4.16.0    True        False         False      29m
machine-approver                           4.16.0    True        False         False      37m
machine-config                             4.16.0    True        False         False      36m
marketplace                                4.16.0    True        False         False      37m
monitoring                                 4.16.0    True        False         False      29m
network                                    4.16.0    True        False         False      38m
node-tuning                                4.16.0    True        False         False      37m
openshift-apiserver                        4.16.0    True        False         False      32m
openshift-controller-manager               4.16.0    True        False         False      30m
openshift-samples                          4.16.0    True        False         False      32m
operator-lifecycle-manager                 4.16.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.16.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.16.0    True        False         False      32m
service-ca                                 4.16.0    True        False         False      38m
storage                                    4.16.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

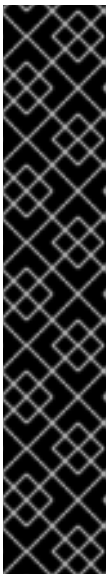
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することを推奨します。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                               1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                               1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running    0          5m
...
```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントで、「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

24.1.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.16 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニタリング を参照してください。

24.1.18. 次のステップ

- クラスターをカスタマイズ します。
- 必要に応じて、リモートヘルスレポートをオプトアウト できます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

第25章 インストール設定

25.1. ノードのカスタマイズ

OpenShift Container Platform は、Ignition を介してクラスター全体の設定とマシンごとの設定の両方をサポートしています。そのため、オペレーティングシステムに対する任意のパーティショニングとファイルコンテンツの変更が可能です。一般に、設定ファイルが Red Hat Enterprise Linux (RHEL) で文書化されている場合、Ignition を介した変更がサポートされます。

マシン設定の変更をデプロイするには 2 つの方法があります。

- **openshift-install** の実行時にクラスターを起動するためにマニフェストファイルに組み込まれるマシン設定を作成します。
- Machine Config Operator を使用して実行中の OpenShift Container Platform ノードに渡されるマシン設定を作成します。

さらに、ベアメタルノードのインストール時に **coreos-installer** に渡される Ignition 設定などの参照設定を変更すると、マシンごとの設定が可能になります。現在、これらの変更はマシン設定オペレーターに表示されません。

以下のセクションでは、この方法でノード上で設定する必要が生じる可能性のある機能について説明します。

25.1.1. Butane でのマシン設定の作成

マシン設定は、ユーザーおよびファイルシステムの作成、ネットワークの設定、systemd ユニットのインストールなどを行う方法をマシンに指示することで、コントロールプレーンマシンおよびワーカーマシンを設定するために使用されます。

マシン設定の変更は困難である可能性があるため、Butane 設定を使用してマシン設定を作成することができます。これにより、ノードの設定がより容易になります。

25.1.1.1. Butane について

Butane は、OpenShift Container Platform が使用するコマンドラインユーティリティーで、マシン設定を作成するための便利で簡略化した構文を提供したり、マシン設定の追加検証を実行したりします。Butane が受け入れる Butane 設定ファイルの形式は、OpenShift Butane config spec で定義されています。

25.1.1.2. Butane のインストール

Butane ツール (**butane**) をインストールして、コマンドラインインターフェイスから OpenShift Container Platform マシン設定を作成できます。対応するバイナリーファイルをダウンロードし、Linux、Windows、または macOS に **butane** をインストールできます。

ヒント

Butane リリースは、古いリリースと、Fedora CoreOS Config Transpiler (FCCT) との後方互換性があります。

手順

1. Butane イメージのダウンロードページ (https://mirror.openshift.com/pub/openshift-v4/clients/butane/) に移動してください。
2. **butane** バイナリーを取得します。
   a. 最新バージョンの Butane の場合は、最新の **butane** イメージを現在のディレクトリーに保存します。

```
$ curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane --output butane
```

   b. オプション: aarch64 や ppc64le など、Butane をインストールする特定のタイプのアーキテクチャーの場合は、適切な URL を指定してください。以下に例を示します。

```
$ curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane-aarch64 --output butane
```

3. ダウンロード済みのバイナリーファイルを実行可能にします。

```
$ chmod +x butane
```

4. **butane** バイナリーファイルを **PATH** にあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証手順

- **butane** コマンドを実行して、Butane ツールを使用できるようになりました。

```
$ butane <butane_file>
```

25.1.1.3. Butane を使用した MachineConfig オブジェクトの作成

Butane を使用して **MachineConfig** オブジェクトを作成できるため、インストール時に、または Machine Config Operator を使用して、ワーカーノードまたはコントロールプレーンノードを設定できます。

前提条件

- **butane** ユーティリティーをインストールしている。

手順

1. Butane 設定ファイルを作成します。以下の例では、**99-worker-custom.bu** という名前のファイルを作成します。このファイルは、カーネルデバッグメッセージを表示するようにシステムコンソールを設定し、chrony タイムサービスのカスタム設定を指定します。

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-custom
  labels:
```

```
    machineconfiguration.openshift.io/role: worker
openshift:
  kernel_arguments:
    - loglevel=7
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          pool 0.rhel.pool.ntp.org iburst
          driftfile /var/lib/chrony/drift
          makestep 1.0 3
          rtcsync
          logdir /var/log/chrony
```

注記

99-worker-custom.bu ファイルは、ワーカーノードのマシン設定を作成するように設定されます。コントロールプレーンノードにデプロイするには、ロールを **worker** から **master** に変更します。どちらの方法でも、デプロイメントの種類ごとに異なるファイル名を使用して手順全体を繰り返すことができます。

2. 直前の手順で作成したファイルを Butane に指定して **MachineConfig** オブジェクトを作成します。

```
$ butane 99-worker-custom.bu -o ./99-worker-custom.yaml
```

MachineConfig オブジェクト YAML ファイルは、マシンの設定を終了するために作成されます。

3. 将来的に **MachineConfig** オブジェクトを更新する必要がある場合に備えて、Butane 設定を保存します。

4. クラスターがまだ起動していない場合は、マニフェストファイルを生成し、**MachineConfig** オブジェクト YAML ファイルを **openshift** ディレクトリーに追加します。クラスターがすでに実行中の場合は、ファイルを以下のように適用します。

```
$ oc create -f 99-worker-custom.yaml
```

関連情報

- カーネルモジュールのノードへの追加
- インストール時のディスクの暗号化およびミラーリング

25.1.2. day-1 カーネル引数の追加

多くの場合、カーネル引数を day-2 アクティビティーとして変更することが推奨されますが、初期クラスターのインストール時にすべてのマスターまたはワーカーノードにカーネル引数を追加することができます。以下は、クラスターのインストール時にカーネル引数を追加して、システムの初回起動前に有効にする必要が生じる可能性のある理由です。

- システムの起動前に、低レベルのネットワーク設定を実行する必要がある場合。
- SELinux などの機能を無効にし、初回起動時にシステムに影響を与えないようにする必要がある場合。

警告

実稼働環境の RHCOS での SELinux の無効化はサポートされていません。ノード上で SELinux が無効になったら、再プロビジョニングしてから実稼働クラスターに再び追加する必要があります。

カーネル引数をマスターまたはワーカーノードに追加するには、**MachineConfig** オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入することができます。

起動時に RHEL 8 カーネルに渡すことのできる引数のリストについては、Kernel.org カーネルパラメーター を参照してください。カーネル引数が OpenShift Container Platform の初回インストールを完了するために必要な場合は、この手順でカーネル引数のみを追加することが推奨されます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. カーネル引数をワーカーまたコントロールプレーンノードに追加するかどうかを決定します。
3. **openshift** ディレクトリーでファイル (例: **99-openshift-machineconfig-master-kargs.yaml**) を作成し、カーネル設定を追加するために **MachineConfig** オブジェクトを定義します。この例では、**loglevel=7** カーネル引数をコントロールプレーンノードに追加します。

```
$ cat << EOF > 99-openshift-machineconfig-master-kargs.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
EOF
```

カーネル引数をワーカーノードに追加する場合は、**master** を **worker** に切り替えます。マスターおよびワーカーノードの両方に追加するために別々の YAML ファイルを作成します。

クラスターの作成を継続できます。

25.1.3. カーネルモジュールのノードへの追加

大半の一般的なハードウェアの場合、Linux カーネルには、コンピューターの起動時にそのハードウェアを使用するために必要となるデバイスドライバーモジュールが含まれます。ただし、一部のハードウェアの場合、Linux でモジュールを利用できません。したがって、各ホストコンピューターにこれらのモジュールを提供する方法を確保する必要があります。この手順では、OpenShift Container Platform クラスターのノードについてこれを実行する方法を説明します。

この手順に従ってカーネルモジュールを最初にデプロイする際、モジュールは現行のカーネルに対して利用可能になります。新規カーネルがインストールされると、kmods-via-containers ソフトウェアはモジュールを再ビルドし、デプロイしてそのモジュールの新規カーネルと互換性のあるバージョンが利用可能になるようにします。

この機能によって各ノードでモジュールが最新の状態に保てるようにするために、以下が実行されます。

- 新規カーネルがインストールされているかどうかを検出するために、システムの起動時に起動する各ノードに systemd サービスを追加します。
- 新規カーネルが検出されると、サービスはモジュールを再ビルドし、これをカーネルにインストールします。

この手順に必要なソフトウェアの詳細については、kmods-via-containers github サイトを参照してください。

以下の重要な点に留意してください。

- この手順はテクノロジープレビューです。
- ソフトウェアのツールおよびサンプルは公式の RPM 形式で利用できず、現時点ではこの手順に記載されている非公式の **github.com** サイトからしか取得できません。
- この手順で追加する必要がある可能性のあるサードパーティーのカーネルモジュールについては Red Hat はサポートしません。
- この手順では、カーネルモジュールのビルドに必要なソフトウェアは RHEL 8 コンテナーにデプロイされます。モジュールは、ノードが新規カーネルを取得する際に各ノードで自動的に再ビルドされることに注意してください。このため、各ノードには、モジュールの再ビルドに必要なカーネルと関連パッケージを含む **yum** リポジトリーへのアクセスが必要です。このコンテンツは、有効な RHEL サブスクリプションを使用して効果的に利用できます。

25.1.3.1. カーネルモジュールコンテナーのビルドおよびテスト

カーネルモジュールを OpenShift Container Platform クラスターにデプロイする前に、プロセスを別の RHEL システムでテストできます。カーネルモジュールのソースコード、KVC フレームワーク、および kmod-via-containers ソフトウェアを収集します。次にモジュールをビルドし、テストします。RHEL 8 システムでこれを行うには、以下を実行します。

手順

1. RHEL 8 システムを登録します。

```
# subscription-manager register
```

2. RHEL 8 システムにサブスクリプションを割り当てます。

```
# subscription-manager attach --auto
```

3. ソフトウェアとコンテナーのビルドに必要なソフトウェアをインストールします。

```
# yum install podman make git -y
```

4. **kmod-via-containers** リポジトリーのクローンを作成します。

   a. リポジトリーのフォルダーを作成します。

```
$ mkdir kmods; cd kmods
```

   b. リポジトリーのクローンを作成します。

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

5. RHEL 8 ビルドホストに KVC フレームワークインスタンスをインストールし、モジュールをテストします。これにより、**kmods-via-container** systemd サービスが追加され、読み込まれます。

   a. **kmod-via-containers** ディレクトリーに移動します。

```
$ cd kmods-via-containers/
```

   b. KVC フレームワークインスタンスをインストールします。

```
$ sudo make install
```

   c. systemd マネージャー設定を再読み込みします。

```
$ sudo systemctl daemon-reload
```

6. カーネルモジュールのソースコードを取得します。ソースコードは、制御下になく、他から提供されるサードパーティーモジュールをビルドするために使用される可能性があります。システムに対してクローン作成できる以下の **kvc-simple-kmod** サンプルのコンテンツと同様のコンテンツが必要になります。

```
$ cd .. ; git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

7. この例では、設定ファイル **simple-kmod.conf** を編集し、Dockerfile の名前を **Dockerfile.rhel** に変更します。

   a. **kvc-simple-kmod** ディレクトリーに移動します。

```
$ cd kvc-simple-kmod
```

   b. Dockerfile の名前を変更します。

```
$ cat simple-kmod.conf
```

   Dockerfile の例

```
KMOD_CONTAINER_BUILD_CONTEXT="https://github.com/kmods-via-containers/kvc-simple-kmod.git"
KMOD_CONTAINER_BUILD_FILE=Dockerfile.rhel
```

```
KMOD_SOFTWARE_VERSION=dd1a7d4
KMOD_NAMES="simple-kmod simple-procfs-kmod"
```

8. この例ではカーネルモジュール **simple-kmod** の **kmods-via-containers@.service** のインスタンスを作成します。

```
$ sudo make install
```

9. **kmods-via-containers@.service** インスタンスを有効にします。

```
$ sudo kmods-via-containers build simple-kmod $(uname -r)
```

10. systemd サービスを有効にし、起動します。

```
$ sudo systemctl enable kmods-via-containers@simple-kmod.service --now
```

a. サービスのステータスを確認します。

```
$ sudo systemctl status kmods-via-containers@simple-kmod.service
```

出力例

```
● kmods-via-containers@simple-kmod.service - Kmods Via Containers - simple-kmod
  Loaded: loaded (/etc/systemd/system/kmods-via-containers@.service;
        enabled; vendor preset: disabled)
  Active: active (exited) since Sun 2020-01-12 23:49:49 EST; 5s ago...
```

11. カーネルモジュールがロードされていることを確認するには、**lsmod** コマンドを使用してモジュールをリスト表示します。

```
$ lsmod | grep simple_
```

出力例

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

12. オプション。他の方法を使用して **simple-kmod** のサンプルが機能していることを確認します。

- **dmesg** を使用してカーネルリングバッファーで Hello world メッセージを探します。

```
$ dmesg | grep 'Hello world'
```

出力例

```
[ 6420.761332] Hello world from simple_kmod.
```

- /**proc** で **simple-procfs-kmod** の値を確認します。

```
$ sudo cat /proc/simple-procfs-kmod
```

出力例

```
simple-procfs-kmod number = 0
```

- **spkut** コマンドを実行して、モジュールの詳細情報を取得します。

```
$ sudo spkut 44
```

出力例

```
KVC: wrapper simple-kmod for 4.18.0-147.3.1.el8_1.x86_64
Running userspace wrapper using the kernel module container...
+ podman run -i --rm --privileged
  simple-kmod-dd1a7d4:4.18.0-147.3.1.el8_1.x86_64 spkut 44
simple-procfs-kmod number = 0
simple-procfs-kmod number = 44
```

その後は、システムの起動時に、このサービスは新規カーネルが実行中であるかどうかをチェックします。新規カーネルがある場合は、サービスは新規バージョンのカーネルモジュールをビルドし、これをロードします。モジュールがすでにビルドされている場合は、これをロードします。

25.1.3.2. カーネルモジュールの OpenShift Container Platform へのプロビジョニング

OpenShift Container Platform クラスターの初回起動時にカーネルモジュールを有効にする必要があるかどうかに応じて、以下のいずれかの方法でデプロイするようにカーネルモジュールを設定できます。

- **クラスターインストール時のカーネルモジュールのプロビジョニング (day-1)**: コンテンツを **MachineConfig** として作成し、これをマニフェストファイルのセットと共に組み込み、これを **openshift-install** に提供できます。
- **Machine Config Operator によるカーネルモジュールのプロビジョニング (day-2)**: カーネルモジュールを追加する際にクラスターが稼働するまで待機できる場合、Machine Config Operator (MCO) を使用してカーネルモジュールソフトウェアをデプロイできます。

いずれの場合も、各ノードは、新規カーネルの検出時にカーネルパッケージと関連ソフトウェアパッケージを取得できる必要があります。該当するコンテンツを取得できるように各ノードをセットアップする方法はいくつかあります。

- 各ノードに RHEL エンタイトルメントを提供します。
- **/etc/pki/entitlement** ディレクトリーから、既存 RHEL ホストの RHEL エンタイトルメントを取得し、それらを Ignition 設定の作成時に提供する他のファイルと同じ場所にコピーします。
- Dockerfile 内で、カーネルおよびその他のパッケージを含む **yum** リポジトリーへのポインターを追加します。これには、新たにインストールされたカーネルと一致させる必要があるため、新規のカーネルパッケージが含まれている必要があります。

25.1.3.2.1. MachineConfig オブジェクトを介したカーネルモジュールのプロビジョニング

MachineConfig オブジェクト でカーネルモジュールソフトウェアをパッケージ化することで、そのソフトウェアをインストール時に、または Machine Config Operator を使用して、ワーカーノードまたはコントロールプレーンノードに配信できます。

手順

1. RHEL 8 システムを登録します。

```
# subscription-manager register
```

2. RHEL 8 システムにサブスクリプションを割り当てます。

```
# subscription-manager attach --auto
```

3. ソフトウェアのビルドに必要なソフトウェアをインストールします。

```
# yum install podman make git -y
```

4. カーネルモジュールおよびツールをホストするディレクトリーを作成します。

```
$ mkdir kmods; cd kmods
```

5. **kmods-via-containers** ソフトウェアを取得します。

   a. **kmods-via-containers** リポジトリーのクローンを作成します。

   ```
   $ git clone https://github.com/kmods-via-containers/kmods-via-containers
   ```

   b. **kvc-simple-kmod** リポジトリーのクローンを作成します。

   ```
   $ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
   ```

6. モジュールソフトウェアを取得します。この例では、**kvc-simple-kmod** が使用されます。

7. fakeroot ディレクトリーを作成し、先にクローン作成したリポジトリーを使用して Ignition で配信するファイルを使用してこれを設定します。

   a. ディレクトリーを作成します。

   ```
   $ FAKEROOT=$(mktemp -d)
   ```

   b. **kmod-via-containers** ディレクトリーに移動します。

   ```
   $ cd kmods-via-containers
   ```

   c. KVC フレームワークインスタンスをインストールします。

   ```
   $ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
   ```

   d. **kvc-simple-kmod** ディレクトリーに移動します。

   ```
   $ cd ../kvc-simple-kmod
   ```

   e. インスタンスを作成します。

   ```
   $ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
   ```

8. fakeroot ディレクトリーのクローンを作成し、以下のコマンドを実行してシンボリックリンク

8. fakeroot ディレクトリーのクローンを作成し、以下のコマンドを実行してシンボリックリンクをターゲットのコピーに置き換えます。

```
$ cd .. && rm -rf kmod-tree && cp -Lpr ${FAKEROOT} kmod-tree
```

9. カーネルモジュールツリーを埋め込む Butane 設定ファイル (**99-simple-kmod.bu**) を作成し、systemd サービスを有効にします。

注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-simple-kmod
  labels:
    machineconfiguration.openshift.io/role: worker 1
storage:
  trees:
    - local: kmod-tree
systemd:
  units:
    - name: kmods-via-containers@simple-kmod.service
      enabled: true
```

1 コントロールプレーンノードでデプロイするには、**worker** を **master** に変更します。コントロールプレーンおよびワーカーノードの両方にデプロイするには、それぞれのノードのタイプに対してこれらの残りの手順を 1 回ずつ実行します。

10. Butane を使用して、配信されるファイルおよび設定を含むマシン設定 YAML ファイルの **99-simple-kmod.yaml** を生成します。

```
$ butane 99-simple-kmod.bu --files-dir . -o 99-simple-kmod.yaml
```

11. クラスターがまだ起動していない場合は、マニフェストファイルを生成し、そのファイルを **openshift** ディレクトリーに追加します。クラスターがすでに実行中の場合は、ファイルを以下のように適用します。

```
$ oc create -f 99-simple-kmod.yaml
```

ノードは **kmods-via-containers@simple-kmod.service** サービスを起動し、カーネルモジュールがロードされます。

12. カーネルモジュールがロードされていることを確認するには、ノードにログインすることができます (**oc debug node/<openshift-node>** を使用してから **chroot /host** を使用します)。モジュールをリスト表示するには、**lsmod** コマンドを使用します。

```
$ lsmod | grep simple_
```

出力例

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

25.1.4. インストール時のディスクの暗号化およびミラーリング

OpenShift Container Platform のインストール時に、クラスターノードでブートディスクの暗号化およびミラーリングを有効にできます。

25.1.4.1. ディスクの暗号化について

インストール時に、コントロールプレーンおよびコンピュートノードのブートディスクの暗号化を有効できます。OpenShift Container Platform は Trusted Platform Module (TPM) v2 および Tang 暗号化モードをサポートします。

TPM v2

これは優先モードです。TPM v2 は、パスフレーズをサーバー上の安全な暗号プロセッサーに保存します。このモードを使用すると、ディスクがサーバーから取り外された場合に、クラスターノード上のブートディスクデータの暗号化が解除されないようにすることができます。

Tang

Tang および Clevis は、ネットワークバインドディスク暗号化 (NBDE) を有効にするサーバーおよびクライアントコンポーネントです。クラスターノードのブートディスクデータを 1 つまたは複数の Tang サーバーにバインドできます。これにより、ノードが Tang サーバーにアクセスできる安全なネットワーク上にない限り、データの復号化が防止されます。Clevis は、クライアント側の復号化の実装に使用される自動復号化フレームワークです。

重要

Tang 暗号化モードを使用したディスクの暗号化は、ユーザーによってプロビジョニングされるインフラストラクチャーでのベアメタルおよび vSphere インストールでのみサポートされます。

以前のバージョンの Red Hat Enterprise Linux CoreOS (RHCOS) では、ディスク暗号化は Ignition 設定で **/etc/clevis.json** を指定して設定されました。このファイルは、OpenShift Container Platform 4.7 以降で作成されたクラスターではサポートされていません。次の手順を使用して、ディスク暗号化を設定します。

TPM v2 または Tang 暗号化モードを有効にすると、RHCOS ブートディスクは LUKS2 形式を使用して暗号化されます。

この機能には以下の特徴があります。

- インストーラーによってプロビジョニングされたインフラストラクチャー、ユーザーによってプロビジョニングされたインフラストラクチャー、および支援されたインストーラーのデプロイメントで利用可能
- アシステッドインストーラーのデプロイメントの場合:
  - 各クラスターは、Tang または TPM の 1 つの暗号化方式のみを持つことができます
  - 一部またはすべてのノードで暗号化を有効にできます
  - Tang のしきい値はありません。すべてのサーバーが有効で動作している必要があります

    - 暗号化はインストールディスクのみに適用され、ワークロードディスクには適用されません

- Red Hat Enterprise Linux CoreOS (RHCOS) システムのみでサポートされる。
- マニフェストのインストール段階でディスク暗号化を設定し、最初の起動以降、ディスクに書き込まれるすべてのデータを暗号化します
- パスフレーズを提供するのにユーザーの介入を必要としない。
- FIPS モードが有効な場合は、AES-256-XTS 暗号化、または AES-256-CBC を使用します。

25.1.4.1.1. 暗号化しきい値の設定

OpenShift Container Platform では、複数の Tang サーバーの要件を指定できます。TPM v2 と Tang 暗号化モードを同時に設定することもできます。これにより、TPM セキュア暗号プロセッサーが存在し、Tang サーバーがセキュアネットワーク経由でアクセスできる場合にのみ、ブートディスクデータの復号化が有効になります。

ブタン設定で **threshold** 属性を使用して、復号化が発生するために必要な TPM v2 および Tang 暗号化条件の最小数を定義できます。

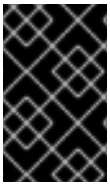
宣言条件の組み合わせで指定値に到達した場合に、しきい値が満たされます。オフラインプロビジョニングの場合、オフラインサーバーは含まれているアドバタイズメントを使用してアクセスされ、オンラインサーバーの数が設定されたしきい値を満たさない場合にのみ、その提供されたアドバタイズメントを使用します。

たとえば、次の設定の **threshold 2** は、オフラインサーバーをバックアップとして使用して 2 つの Tang サーバーにアクセスするか、TPM セキュア暗号プロセッサーといずれかの Tang サーバーにアクセスすることによって到達できます。

ディスク暗号化の Butane 設定例

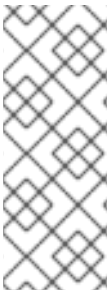
```
variant: openshift
version: 4.16.0
metadata:
  name: worker-storage
  labels:
    machineconfiguration.openshift.io/role: worker
boot_device:
  layout: x86_64 (1)
  luks:
    tpm2: true (2)
    tang: (3)
      - url: http://tang1.example.com:7500
        thumbprint: jwGN5tRFK-kF6pIX89ssF3khxxX
      - url: http://tang2.example.com:7500
        thumbprint: VCJsvZFjBSIHSldw78rOrq7h2ZF
      - url: http://tang3.example.com:7500
        thumbprint: PLjNyRdGw03zlRoGjQYMahSZGu9
        advertisement: "{\"payload\": \"...\", \"protected\": \"...\", \"signature\": \"...\"}" (4)
    threshold: 2 (5)
openshift:
  fips: true
```

1. このフィールドをクラスターノードの命令セットアーキテクチャーに設定します。いくつかの例には、**x86_64**、**aarch64**、または **ppc64le** が含まれます。
2. Trusted Platform Module (TPM) を使用してルートファイルシステムを暗号化する場合は、このフィールドを追加してください。
3. 1 台以上の Tang サーバーを使用する必要がある場合は、このセクションを追加してください。
4. オプション: オフラインプロビジョニングにこのフィールドを含めます。Ignition は、実行時にサーバーからアドバタイズメントを取得するのではなく、Tang サーバーバインディングをプロビジョニングします。これにより、プロビジョニング時にサーバーが利用できなくなります。
5. 復号化を行うために必要な TPM v2 および Tang 暗号化条件の最小数を指定します。



重要

デフォルトの **しきい値** は **1** です。設定に複数の暗号化条件を追加するにも拘らず、しきい値を指定しない場合は、条件のいずれかが満たされている場合に復号化を実行できます。



注記

復号化に TPM v2 と Tang が必要な場合、**threshold** 属性の値は、指定された Tang サーバーの総数に 1 を加えた数に等しくなければなりません。**threshold** が低い場合は、単一の暗号化モードを使用してしきい値に達する可能性があります。たとえば、**tpm2** を **true** に設定し、2 つの Tang サーバーを指定した場合、TPM セキュア暗号プロセッサーが利用できない場合でも、2 つの Tang サーバーにアクセスすることでしきい値 **2** を満たすことができます。

25.1.4.2. ディスクのミラーリングについて

コントロールプレーンおよびワーカーノードでの OpenShift Container Platform のインストール時に、ブートおよびその他のディスクの 2 つ以上の冗長ストレージデバイスへのミラーリングを有効にできます。1 つのデバイスが使用可能なままであれば、ストレージデバイスの障害後もノードは機能し続けます。

ミラーリングは、障害の発生したディスクの置き換えをサポートしません。ノードを再プロビジョニングして、ミラーを本来の劣化していない状態に復元します。

注記

ユーザーがプロビジョニングしたインフラストラクチャーのデプロイメントの場合、ミラーリングは RHCOS システムのみで使用できます。ミラーリングのサポートは、BIOS または UEFI で起動された **x86_64** ノードおよび **ppc64le** ノードで利用できます。

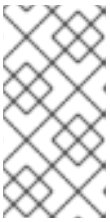
25.1.4.3. ディスク暗号化およびミラーリングの設定

OpenShift Container Platform のインストール時に暗号化およびミラーリングを有効にし、設定することができます。

前提条件

- インストールノードで OpenShift Container Platform インストールプログラムをダウンロードしている。

- インストールノードに Butane がインストールされている。



注記

Butane は、OpenShift Container Platform が使用するコマンドラインユーティリティーであり、マシンの設定を記述および検証するための便利で簡潔な構文を提供します。詳細は、Butane を使用したマシン設定の作成を参照してください。

- Tang 交換キーのサムプリントの生成に使用できる Red Hat Enterprise Linux (RHEL) 8 マシンにアクセスできる。

手順

1. TPM v2 を使用してクラスターを暗号化する必要がある場合、TPM v2 暗号化を各ノードのホストファームウェアで有効にする必要があるかどうかを確認します。これは、ほとんどの Dell システムで必要になります。特定のシステムのマニュアルを確認してください。
2. Tang を使用してクラスターを暗号化する必要がある場合は、以下の準備段階の手順に従います。
   a. Tang サーバーを設定するか、既存のサーバーにアクセスします。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。
   b. RHEL 8 マシンに **clevis** パッケージがインストールされていない場合はインストールします。

```
$ sudo yum install clevis
```

   c. RHEL 8 マシンで以下のコマンドを実行し、交換キーのサムプリントを生成します。**http://tang1.example.com:7500** を Tang サーバーの URL に置き換えます。

```
$ clevis-encrypt-tang '{"url":"http://tang1.example.com:7500"}' < /dev/null > /dev/null (1)
```

(1) この例では、**tangd.socket** は Tang サーバーのポート **7500** でリッスンしています。

注記

clevis-encrypt-tang コマンドは、交換キーの拇印を生成します。このステップでは、データは暗号化コマンドに渡されません。**/dev/null** は、プレーンテキストの代わりに入力としてここに存在します。この手順には必要ないため、暗号化された出力は **/dev/null** に送信されます。

出力例

```
The advertisement contains the following signing keys:

PLjNyRdGw03zlRoGjQYMahSZGu9 (1)
```

(1) エクスチェンジキーのサムプリント。

Do you want to trust these keys? [ynYN] のプロンプトが表示されたら、**Y** と入力します。

d. オプション: オフライン Tang プロビジョニングの場合は、以下を行います。

i. **curl** コマンドを使用して、サーバーからアドバタイズメントを取得します。**http://tang2.example.com:7500** を Tang サーバーの URL に置き換えます。

```
$ curl -f http://tang2.example.com:7500/adv > adv.jws && cat adv.jws
```

予想される出力

```
{"payload": "eyJrZXlzIjogW3siYWxnIjogIkV", "protected":
"eyJhbGciOiJFUzUxMiIsImN0eSI", "signature": "ADLgk7fZdE3Yt4FyYsm0pHiau7Q"}
```

ii. 暗号化のためにアドバタイズメントファイルを Clevis に提供します。

```
$ clevis-encrypt-tang '{"url":"http://tang2.example.com:7500","adv":"adv.jws"}' <
/dev/null > /dev/null
```

e. ノードが静的 IP アドレス指定で設定されている場合は、RHCOS ノードをインストールするときに **coreos-installer iso customize --dest-karg-append** を実行するか、**coreos-installer --append-karg** オプションを使用して、インストール済みシステムの IP アドレスを設定します。ネットワークに必要な **ip=** およびその他の引数を追加します。

重要

一部の静的 IP の設定方法は、初回のブート後に initramfs に影響を与えず、Tang 暗号化では機能しない場合があります。これらには、**coreos-installer --copy-network** オプション、**coreos-installer iso customize --network-keyfile** オプション、および **coreos-installer pxe customize --network-keyfile** オプションが含まれるほか、インストール中のライブ ISO または PXE イメージのカーネルコマンドラインに **ip=** 引数が追加されます。静的 IP 設定が間違っていると、ノードの 2 回目のブートが失敗します。

3. インストールノードで、インストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、インストールファイルを保存するディレクトリーへのパスに置き換えます。

4. ディスクの暗号化、ミラーリング、またはそれら両方を設定する Butane 設定を作成します。たとえば、コンピュートノードのストレージを設定するには、**$HOME/clusterconfig/worker-storage.bu** ファイルを作成します。

起動デバイスの Butane 設定例

```
variant: openshift
version: 4.16.0
metadata:
  name: worker-storage 1
```

```
  labels:
    machineconfiguration.openshift.io/role: worker (2)
boot_device:
  layout: x86_64 (3)
  luks: (4)
    tpm2: true (5)
    tang: (6)
      - url: http://tang1.example.com:7500 (7)
        thumbprint: PLjNyRdGw03zlRoGjQYMahSZGu9 (8)
      - url: http://tang2.example.com:7500
        thumbprint: VCJsvZFjBSIHSldw78rOrq7h2ZF
        advertisement: "{"payload": "eyJrZXlzIjogW3siYWxnIjogIkV", "protected":
"eyJhbGciOiJFUzUxMiIsImN0eSI", "signature": "ADLgk7fZdE3Yt4FyYsm0pHiau7Q"}" (9)
    threshold: 1 (10)
  mirror: (11)
    devices: (12)
      - /dev/sda
      - /dev/sdb
openshift:
  fips: true (13)
```

(1) (2) コントロールプレーンの設定については、これらの両方の場所で **worker** を **master** に置き換えます。

(3) このフィールドをクラスターノードの命令セットアーキテクチャーに設定します。いくつかの例には、**x86_64**、**aarch64**、または **ppc64le** が含まれます。

(4) ルートファイルシステムを暗号化する必要がある場合は、このセクションを追加してください。詳細は、ディスクの暗号化についてを参照してください。

(5) Trusted Platform Module (TPM) を使用してルートファイルシステムを暗号化する場合は、このフィールドを追加してください。

(6) 1 台以上の Tang サーバーを使用する必要がある場合は、このセクションを追加してください。

(7) Tang サーバーの URL を指定します。この例では、**tangd.socket** は Tang サーバーのポート **7500** でリッスンしています。

(8) 前述のステップで生成された Exchange キーサムプリントを指定します。

(9) オプション: オフライン Tang サーバーのアドバタイズメントを有効な JSON 形式で指定します。

(10) 復号化に実行に必要な TPM v2 および Tang 暗号化条件の最小数を指定します。デフォルト値は **1** です。このトピックについて詳しくは、暗号化しきい値の設定を参照してください。

(11) ブートディスクをミラーリングする必要がある場合は、このセクションを追加してください。詳細は、ディスクのミラーリングについてを参照してください。

(12) RHCOS がインストールされるディスクを含む、ブートディスクミラーに含まれる必要があるすべてのディスクデバイスを一覧表示します。

(13) クラスターで FIPS モードを有効にするためにこのディレクティブを追加します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。ノードをディスク暗号化とミラーリングの両方を使用するように設定する場合、両方の機能を同じ Butane 設定ファイルに設定する必要があります。FIPS モードが有効にされたノードでディスク暗号化を設定する場合は、FIPS モードが別のマニフェストで有効化されている場合でも、同じ Butane 設定ファイルに **fips** ディレクティブを追加する必要があります。

5. 対応する Butane 設定ファイルからコントロールプレーンまたはコンピュートノードのマニフェストを作成し、**<installation_directory>/openshift** ディレクトリーに保存します。たとえば、コンピュートノードのマニフェストを作成するには、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/worker-storage.bu -o <installation_directory>/openshift/99-worker-storage.yaml
```

   ディスクの暗号化またはミラーリングを必要とするノード種別ごとに、この手順を繰り返します。

6. 今後マニフェストを更新する必要がある場合には、Butane 設定ファイルを保存します。

7. 残りの OpenShift Container Platform インストールを継続します。

ヒント

インストール時に、ディスク暗号化またはミラーリングに関連するエラーメッセージがないか、RHCOS ノードでコンソールログをモニタリングできます。

重要

追加のデータパーティションを設定する場合、暗号化が明示的に要求されない限り、それらは暗号化されません。

検証

OpenShift Container Platform のインストール後に、ブートディスクの暗号化またはミラーリングがクラスターノードで有効にされているかどうかを確認できます。

1. インストールホストから、デバッグ Pod を使用してクラスターノードにアクセスします。

   a. ノードのデバッグ Pod を開始します。次に例を示します。

```
$ oc debug node/compute-1
```

   b. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にノードのルートファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ノードの実行可能パスに含まれるバイナリーを実行できます。

```
# chroot /host
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、**kubelet** がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

2. ブートディスクの暗号化を設定している場合は、有効であるかどうかを確認します。

   a. デバッグシェルで、ノードでのルートマッピングのステータスを確認します。

```
# cryptsetup status root
```

出力例

```
/dev/mapper/root is active and is in use.
  type:    LUKS2 (1)
  cipher:  aes-xts-plain64 (2)
  keysize: 512 bits
  key location: keyring
  device:  /dev/sda4 (3)
  sector size:  512
  offset:  32768 sectors
  size:    15683456 sectors
  mode:    read/write
```

(1) 暗号化形式。TPM v2 または Tang 暗号化モードを有効にすると、RHCOS ブートディスクは LUKS2 形式を使用して暗号化されます。

(2) LUKS2 ボリュームの暗号化に使用される暗号化アルゴリズム。FIPS モードが有効な場合には 、**aes-cbc-essiv:sha256** 暗号が使用されます。

(3) 暗号化した LUKS2 ボリュームを含むデバイス。ミラーリングを有効にすると、値は **/dev/md126** などのソフトウェアミラーデバイスを表します。

   b. 暗号化されたデバイスにバインドされる Clevis プラグインを一覧表示します。

```
# clevis luks list -d /dev/sda4 (1)
```

(1) 前述のステップの出力の **device** フィールドに一覧表示されるデバイスを指定します。

出力例

```
1: sss '{"t":1,"pins":{"tang":[{"url":"http://tang.example.com:7500"}]}}' (1)
```

(1) この出力例では、Tang プラグインは、**/dev/sda4** デバイスの Shamir の Secret Sharing (SSS) Clevis プラグインにより使用されます。

3. ミラーリングを設定している場合は、有効かどうかを確認します。

   a. デバッグシェルから、ノードにあるソフトウェアの RAID デバイスのリストを表示します。

   ```
   # cat /proc/mdstat
   ```

   出力例

   ```
   Personalities : [raid1]
   md126 : active raid1 sdb3[1] sda3[0] (1)
     393152 blocks super 1.0 [2/2] [UU]

   md127 : active raid1 sda4[0] sdb4[1] (2)
     51869632 blocks super 1.2 [2/2] [UU]

   unused devices: <none>
   ```

   (1) **/dev/md126** ソフトウェア RAID ミラーデバイスは、クラスターノードの **/dev/sda3** および **/dev/sdb3** ディスクデバイスを使用します。

   (2) **/dev/md127** ソフトウェア RAID ミラーデバイスは、クラスターノードの **/dev/sda4** および **/dev/sdb4** ディスクデバイスを使用します。

   b. 上記のコマンドの出力に記載されている各ソフトウェア RAID デバイスの詳細を確認してください。以下の例は、**/dev/md126** デバイスの詳細を示しています。

   ```
   # mdadm --detail /dev/md126
   ```

   出力例

   ```
   /dev/md126:
              Version : 1.0
        Creation Time : Wed Jul  7 11:07:36 2021
           Raid Level : raid1 (1)
           Array Size : 393152 (383.94 MiB 402.59 MB)
        Used Dev Size : 393152 (383.94 MiB 402.59 MB)
         Raid Devices : 2
        Total Devices : 2
          Persistence : Superblock is persistent

          Update Time : Wed Jul  7 11:18:24 2021
                State : clean (2)
       Active Devices : 2 (3)
      Working Devices : 2 (4)
       Failed Devices : 0 (5)
        Spare Devices : 0

   Consistency Policy : resync

                 Name : any:md-boot (6)
                 UUID : ccfa3801:c520e0b5:2bee2755:69043055
               Events : 19
   ```

```
    Number   Major   Minor   RaidDevice State
       0     252       3        0      active sync   /dev/sda3 7
       1     252       19       1      active sync   /dev/sdb3 8
```

1 デバイスの RAID レベルを指定します。**raid1** は、RAID 1 ディスクミラーリングを示します。

2 RAID デバイスの状態を指定します。

3 4 アクティブかつ機能している基礎となるディスクデバイスの数を示します。

5 ステータスが failed のディスクデバイスの数を示します。

6 ソフトウェア RAID デバイスの名前。

7 8 ソフトウェア RAID デバイスが使用する基礎となるディスクデバイスに関する情報を提供します。

c. ソフトウェア RAID デバイスにマウントされているファイルシステムを一覧表示します。

```
# mount | grep /dev/md
```

出力例

```
/dev/md127 on / type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /etc type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /usr type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /sysroot type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/containers/storage/overlay type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/1 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/2 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/3 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/4 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/5 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md126 on /boot type ext4 (rw,relatime,seclabel)
```

この出力例では、**/boot** ファイルシステムが **/dev/md126** software RAID デバイスに、root

この出力例では、**/boot** ファイルシステムが **/dev/md126** software RAID デバイスに、root ファイルシステムが **/dev/md127** にマウントされ t ています。

4. OpenShift Container Platform ノードタイプごとに検証手順を繰り返します。

関連情報

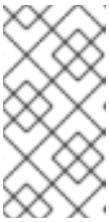
- TPM v2 および Tang 暗号化モードの詳細は、ポリシーベースの複号を使用して暗号化ボリュームの自動アンロックの設定 を参照してください。

25.1.4.4. RAID 対応のデータボリュームの設定

ソフトウェア RAID のパーティション設定を有効にして、外部データボリュームを提供できます。OpenShift Container Platform は、データ保護およびフォールトトレランスに対応するために RAID 0、RAID 1、RAID 4、RAID 5、RAID 6、および RAID 10 をサポートします。詳細は、ディスクのミラーリングについてを参照してください。

前提条件

- インストールノードで OpenShift Container Platform インストールプログラムをダウンロードしている。
- インストールノードに Butane をインストールしている。



注記

Butane は、OpenShift Container Platform が使用するコマンドラインユーティリティーで、マシン設定を作成するための便利で簡略化した構文を提供したり、マシン設定の追加検証を実行したりします。詳細は、**Butane を使用したマシン設定の作成** セクションを参照してください。

手順

1. ソフトウェア RAID を使用してデータボリュームを設定する Butane 設定を作成します。
   - ミラーリングされた起動ディスクに使用されるのと同じディスク上に RAID 1 を使用してデータボリュームを設定するには、**$HOME/clusterconfig/raid1-storage.bu** ファイルを作成します。以下に例を示します。

     ミラーリングされた起動ディスク上の RAID 1

```
variant: openshift
version: 4.16.0
metadata:
  name: raid1-storage
  labels:
    machineconfiguration.openshift.io/role: worker
boot_device:
  mirror:
    devices:
      - /dev/disk/by-id/scsi-3600508b400105e210000900000490000
      - /dev/disk/by-id/scsi-SSEAGATE_ST373453LW_3HW1RHM6
storage:
  disks:
    - device: /dev/disk/by-id/scsi-3600508b400105e210000900000490000
      partitions:
```

```
      - label: root-1
        size_mib: 25000 1
      - label: var-1
  - device: /dev/disk/by-id/scsi-SSEAGATE_ST373453LW_3HW1RHM6
    partitions:
      - label: root-2
        size_mib: 25000 2
      - label: var-2
raid:
  - name: md-var
    level: raid1
    devices:
      - /dev/disk/by-partlabel/var-1
      - /dev/disk/by-partlabel/var-2
filesystems:
  - device: /dev/md/md-var
    path: /var
    format: xfs
    wipe_filesystem: true
    with_mount_unit: true
```

1 2 データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

- セカンダリーディスク上に RAID 1 を使用してデータボリュームを設定するには、**$HOME/clusterconfig/raid1-alt-storage.bu** ファイルを作成します。以下に例を示します。

  セカンダリーディスク上の RAID 1

```
variant: openshift
version: 4.16.0
metadata:
  name: raid1-alt-storage
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  disks:
    - device: /dev/sdc
      wipe_table: true
      partitions:
        - label: data-1
    - device: /dev/sdd
      wipe_table: true
      partitions:
        - label: data-2
  raid:
    - name: md-var-lib-containers
      level: raid1
      devices:
        - /dev/disk/by-partlabel/data-1
```

```
      - /dev/disk/by-partlabel/data-2
filesystems:
  - device: /dev/md/md-var-lib-containers
    path: /var/lib/containers
    format: xfs
    wipe_filesystem: true
    with_mount_unit: true
```

2. 前のステップで作成した Butane 設定から RAID マニフェストを作成し、それを **<installation_directory>/openshift** ディレクトリーに保存します。たとえば、コンピュートノードのマニフェストを作成するには、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/<butane_config>.bu -o
<installation_directory>/openshift/<manifest_name>.yaml 1
```

1 **<butane_config>** および **<manifest_name>** を直前の手順のファイル名に置き換えます。たとえば、セカンダリーディスクの場合は、**raid1-alt-storage.bu** および **raid1-alt-storage.yaml** になります。

3. 今後マニフェストを更新する必要がある場合には、Butane 設定を保存します。

4. 残りの OpenShift Container Platform インストールを継続します。

25.1.4.5. CPU (VROC)データボリュームでの Intel® 仮想 RAID の設定

Intel® VROC はハイブリッド RAID の一種で、一部のメンテナンスはハードウェアにオフロードされますが、ソフトウェア RAID としてオペレーティングシステムに表示されます。

以下の手順では、Intel® VROC が有効な RAID1 を設定します。

前提条件

- Intel® Volume Management Device (VMD)が有効になっているシステムがあります。

手順

1. 次のコマンドを実行して、Intel® Matrix Storage Manager (IMSM) RAID コンテナーを作成します。

```
$ mdadm -CR /dev/md/imsm0 -e \
 imsm -n2 /dev/nvme0n1 /dev/nvme1n1 1
```

1 RAID デバイス名。この例では、2 つのデバイスがリストされています。2 つ以上のデバイス名を指定する場合は、**-n** フラグを調整する必要があります。たとえば、3 つのデバイスを一覧表示すると、flag **-n3** が使用されます。

2. コンテナーに RAID1 ストレージを作成します。

   a. 以下のコマンドを実行して、実際の RAID1 ボリュームの前にダミー RAID0 ボリュームを作成します。

```
$ mdadm -CR /dev/md/dummy -l0 -n2 /dev/imsm0 -z10m --assume-clean
```

b. 次のコマンドを実行して、実際の RAID1 アレイを作成します。

```
$ mdadm -CR /dev/md/coreos -l1 -n2 /dev/imsm0
```

c. 以下のコマンドで、RAID0 と RAID1 の両方のメンバーアレイを停止し、ダミー RAID0 アレイを削除します。

```
$ mdadm -S /dev/md/dummy \
  mdadm -S /dev/md/coreos \
  mdadm --kill-subarray=0 /dev/md/imsm0
```

d. 次のコマンドを実行して、RAID1 アレイを再起動します。

```
$ mdadm -A /dev/md/coreos /dev/md/imsm0
```

3. RAID1 デバイスに RHCOS をインストールします。

a. 次のコマンドを実行して、IMSM コンテナーの UUID を取得します。

```
$ mdadm --details --export /dev/md/imsm0
```

b. 次のコマンドを実行して、RHCOS をインストールし、**rd.md.uuid** カーネル引数を追加します。

```
$ coreos-installer install /dev/md/coreos \
  --append-karg rd.md.uuid=<md_UUID> 1
  ...
```

1 IMSM コンテナーの UUID。

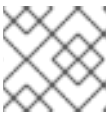
RHCOS をインストールするために必要な追加の **coreos-installer** 引数を含めます。

25.1.5. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定できます。

手順

1. **chrony.conf** ファイルのコンテンツを含む Butane 設定を作成します。たとえば、ワーカーノードで chrony を設定するには、**99-worker-chrony.bu** ファイルを作成します。



注記

Butane の詳細は、"Butane を使用したマシン設定の作成" を参照してください。

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-chrony 1
  labels:
```

```
    machineconfiguration.openshift.io/role: worker (2)
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 (3)
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst (4)
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

(1) (2) コントロールプレーンノードでは、これらの両方の場所で **worker** の代わりに **master** を使用します。

(3) マシン設定ファイルの **mode** フィールドに 8 進数の値でモードを指定します。ファイルを作成し、変更を適用すると、**mode** は 10 進数の値に変換されます。コマンド **oc get mc <mc-name> -o yaml** で YAML ファイルを確認できます。

(4) DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。または、NTP サーバーの **1.rhel.pool.ntp.org**、**2.rhel.pool.ntp.org**、または **3.rhel.pool.ntp.org** のいずれかを指定できます。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-worker-chrony.yaml**) を生成します。

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 以下の 2 つの方法のいずれかで設定を適用します。

   - クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、**MachineConfig** オブジェクトファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を続行します。
   - クラスターがすでに実行中の場合は、ファイルを適用します。

```
$ oc apply -f ./99-worker-chrony.yaml
```

25.1.6. 関連情報

- Butane の詳細は、Butane を使用したマシン設定の作成 を参照してください。
- FIPS サポートの詳細は、FIPS 暗号のサポート を参照してください。

25.2. ファイアウォールの設定

ファイアウォールを使用する場合、OpenShift Container Platform が機能するために必要なサイトにアクセスできるように設定する必要があります。一部のサイトにはアクセスを常に付与し、クラスターをホストするために Red Hat Insights、Telemetry サービス、クラウドを使用したり、特定のビルドストラテジーをホストする場合に追加のアクセスを付与する必要があります。

25.2.1. OpenShift Container Platform のファイアウォールの設定

OpenShift Container Platform をインストールする前に、ファイアウォールを、OpenShift Container Platform が必要とするサイトへのアクセスを付与するように設定する必要があります。ファイアウォールを使用する場合は、OpenShift Container Platform が機能するために必要なサイトにアクセスできるように、ファイアウォールに追加の設定を行います。

ワーカーノードと比較して、コントローラーノードのみで実行されるサービスには、特別な設定上の考慮事項はありません。

注記

ご使用の環境で OpenShift Container Platform クラスターの前に専用のロードバランサーがある場合は、ファイアウォールとロードバランサーの間の許可リストを確認して、クラスターに対する不要なネットワーク制限を回避してください。

手順

1. ファイアウォールの許可リストに次のレジストリー URL を設定します。

URL	ポート	機能
registry.redhat.io	443	コアコンテナーイメージを指定します。
access.redhat.com [1]	443	コアコンテナーイメージを含め、Red Hat Ecosytem Catalog に保存されているすべてのコンテナーイメージをホストします。
quay.io	443	コアコンテナーイメージを指定します。
cdn.quay.io	443	コアコンテナーイメージを指定します。
cdn01.quay.io	443	コアコンテナーイメージを指定します。
cdn02.quay.io	443	コアコンテナーイメージを指定します。
cdn03.quay.io	443	コアコンテナーイメージを指定します。
cdn04.quay.io	443	コアコンテナーイメージを指定します。
cdn05.quay.io	443	コアコンテナーイメージを指定します。
cdn06.quay.io	443	コアコンテナーイメージを指定します。
sso.redhat.com	443	**https://console.redhat.com** サイトは、**sso.redhat.com** からの認証を使用します。

1. ファイアウォール環境では、**access.redhat.com** リソースが許可リストに含まれていることを確認してください。このリソースは、コンテナークライアントが **registry.access.redhat.com** からイメージを取得するときにイメージを検証するために必

要な署名ストアをホストします。

許可リストで **cdn.quay.io** と **cdn0[1-3].quay.io** の代わりに、ワイルドカードの ***.quay.io** と ***.openshiftapps.com** を使用できます。**quay.io** などのサイトを許可リストに追加するには、***.quay.io** などのワイルドカードエントリーを拒否リストに加えないでください。ほとんどの場合、イメージレジストリーはコンテンツ配信ネットワーク (CDN) を使用してイメージを提供します。ファイアウォールがアクセスをブロックすると、最初のダウンロード要求が **cdn01.quay.io** などのホスト名にリダイレクトされるときに、イメージのダウンロードが拒否されます。

2. ファイアウォールの許可リストを設定し、ビルドに必要な言語またはフレームワークのリソースを提供するサイトをリストに含めます。

3. Telemetry を無効にしていない場合は、以下の URL へのアクセスを許可して Red Hat Insights にアクセスできるようにする必要があります。

URL	ポート	機能
cert-api.access.redhat.com	443	Telemetry で必須
api.access.redhat.com	443	Telemetry で必須
infogw.api.openshift.com	443	Telemetry で必須
console.redhat.com	443	Telemetry および **insights-operator** で必須

4. Alibaba Cloud、Amazon Web Services (AWS) 、Microsoft Azure、または Google Cloud Platform (GCP) を使用してクラスターをホストする場合、クラウドプロバイダー API およびそのクラウドの DNS を提供する URL へのアクセス権を付与する必要があります。

クラウド	URL	ポート	機能
Alibaba	***.aliyuncs.com**	443	Alibaba Cloud のサービスとリソースにアクセスするために必要です。Alibaba の endpoints_config.go ファイル を参照して、使用するリージョンを許可する正確なエンドポイントを確認してください。
AWS	**aws.amazon.com**	443	AWS 環境でのクラスターのインストールや管理に使用されます。
	***.amazonaws.com** または、AWS API にワイルドカードを使用しない場合、次の URL を許可リストに含める必要があります。	443	AWS サービスおよびリソースへのアクセスに必要です。AWS ドキュメントの AWS Service Endpoints を参照して、使用するリージョンを許可する正確なエンドポイントを確認してください。

クラウド	URL	ポート	機能
	ec2.amazonaws.com	443	AWS 環境でのクラスターのインストールや管理に使用されます。
	events.amazonaws.com	443	AWS 環境でのクラスターのインストールや管理に使用されます。
	iam.amazonaws.com	443	AWS 環境でのクラスターのインストールや管理に使用されます。
	route53.amazonaws.com	443	AWS 環境でクラスターをインストールし、管理するのに使用されます。
	s3.amazonaws.com	443	AWS 環境でクラスターをインストールし、管理するのに使用されます。
	s3.<aws_region>.amazonaws.com	443	AWS 環境でクラスターをインストールし、管理するのに使用されます。
	s3.dualstack.<aws_region>.amazonaws.com	443	AWS 環境でのクラスターのインストールや管理に使用されます。
	sts.amazonaws.com	443	AWS 環境でクラスターをインストールし、管理するのに使用されます。
	sts.<aws_region>.amazonaws.com	443	AWS 環境でクラスターをインストールし、管理するのに使用されます。
	tagging.us-east-1.amazonaws.com	443	AWS 環境でクラスターをインストールし、管理するのに使用されます。このエンドポイントは、クラスターがデプロイされているリージョンに関係なく、常に **us-east-1** です。
	ec2.<aws_region>.amazonaws.com	443	AWS 環境でのクラスターのインストールや管理に使用されます。
	elasticloadbalancing.<aws_region>.amazonaws.com	443	AWS 環境でのクラスターのインストールや管理に使用されます。
	servicequotas.<aws_region>.amazonaws.com	443	必須。サービスをデプロイするためのクォータを確認するのに使用されます。
	tagging.<aws_region>.amazonaws.com	443	タグの形式で AWS リソースに関するメタデータを割り当てることができます。

クラウド	URL	ポート	機能
	***.cloudfront.net**	443	CloudFront へのアクセスを提供するために使用されます。AWS Security Token Service (STS) およびプライベート S3 バケットを使用する場合は、CloudFront へのアクセスを提供する必要があります。
GCP	***.googleapis.com**	443	GCP サービスおよびリソースへのアクセスに必要です。GCP ドキュメントの Cloud Endpoints を参照して、お使いの API を許可するエンドポイントを確認してください。
	accounts.google.com	443	GCP アカウントへのアクセスに必要です。
Microsoft Azure	**management.azure.com**	443	Microsoft Azure のサービスとリソースにアクセスするために必要です。Microsoft Azure ドキュメントの Microsoft Azure REST API reference を参照して、お使いの API を許可するエンドポイントを確認してください。
	***.blob.core.windows.net**	443	Ignition ファイルのダウンロードに必要です。
	login.microsoftonline.com	443	Microsoft Azure のサービスとリソースにアクセスするために必要です。Microsoft Azure ドキュメントの Azure REST API reference を参照して、お使いの API を許可するエンドポイントを確認してください。

5. 以下の URL を許可リストに指定します。

URL	ポート	機能
mirror.openshift.com	443	ミラーリングされたインストールのコンテンツおよびイメージへのアクセスに必要。Cluster Version Operator には単一の機能ソースのみが必要ですが、このサイトはリリースイメージ署名のソースでもあります。
storage.googleapis.com/openshift-release	443	リリースイメージ署名のソース (ただし、Cluster Version Operator には単一の機能ソースのみが必要)。

URL	ポート	機能
***.apps.<cluster_name>.<base_domain>**	443	Ingress ワイルドカードをインストール時に設定しない限り、デフォルトのクラスタールートへのアクセスに必要。
quayio-production-s3.s3.amazonaws.com	443	AWS で Quay イメージコンテンツにアクセスするために必要。
api.openshift.com	443	クラスタートークンの両方が必要であり、クラスターに更新が利用可能かどうかを確認するために必要です。
rhcos.mirror.openshift.com	443	Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードするために必要。
console.redhat.com	443	クラスタートークンに必須
sso.redhat.com	443	**https://console.redhat.com** サイトは、**sso.redhat.com** からの認証を使用します。

Operator にはヘルスチェックを実行するためのルートアクセスが必要です。具体的には、認証および Web コンソール Operator は 2 つのルートに接続し、ルートが機能することを確認します。クラスター管理者として操作を実行しており、***.apps.<cluster_name>.<base_domain>** を許可しない場合は、これらのルートを許可します。

- **oauth-openshift.apps.<cluster_name>.<base_domain>**
- **console-openshift-console.apps.<cluster_name>.<base_domain>**、またはフィールドが空でない場合に **consoles.operator/cluster** オブジェクトの **spec.route.hostname** フィールドに指定されるホスト名。

6. オプションのサードパーティーコンテンツに対する次の URL を許可リストに追加します。

URL	ポート	機能
registry.connect.redhat.com	443	すべてのサードパーティーのイメージと認定 Operator に必要です。
rhc4tp-prod-z8cxf-image-registry-us-east-1-evenkyleffocxqvofrk.s3.dualstack.us-east-1.amazonaws.com	443	**registry.connect.redhat.com** でホストされているコンテナーイメージにアクセスできます
oso-rhc4tp-docker-registry.s3-us-west-2.amazonaws.com	443	Sonatype Nexus、F5 Big IP Operator に必要です。

7. デフォルトの Red Hat Network Time Protocol (NTP) サーバーを使用する場合は、以下の URL を許可します。
    - **1.rhel.pool.ntp.org**
    - **2.rhel.pool.ntp.org**
    - **3.rhel.pool.ntp.org**

注記

デフォルトの Red Hat NTP サーバーを使用しない場合は、プラットフォームの NTP サーバーを確認し、ファイアウォールでこれを許可します。

関連情報

- AWS STS の OpenID Connect の要件

25.2.2. OpenShift Container Platform ネットワークフローマトリックス

ネットワークフローマトリックスは、OpenShift Container Platform サービスへの Ingress フローを記述します。マトリックスのネットワーク情報は、ベアメタル環境とクラウド環境の両方で正確です。Ingress トラフィックの管理に役立つ、ネットワークフローマトリックスの情報を使用します。Ingress トラフィックを重要なフローに制限して、ネットワークセキュリティーを向上させることができます。

raw の CSV コンテンツを表示またはダウンロードするには、このリソース を参照してください。

さらに、イングレストラフィックを管理するときは、次の動的ポート範囲を検討してください。

- **9000-9999**: ホストレベルのサービス
- Kubernetes ノードポート
- **49152-65535**: 動的ポートまたはプライベートポート

注記

ネットワークフローマトリックスは、ベース OpenShift Container Platform インストールの ingress トラフィックフローを記述します。Red Hat Marketplace から入手可能なオプション Operator などの追加コンポーネントのネットワークフローは記述されません。このマトリックスは、Hosted-Control-Plane、MicroShift、またはスタンドアロンクラスターには適用されません。

表25.1 ネットワークフローマトリックス

方向	プロトコル	ポート	namespace	Service	Pod	Container	Node Role	任意
Ingress	TCP	22	ホストシステムサービス	sshd			master	TRUE
Ingress	TCP	53	openshift-dns	dns-default	dnf-default	dns	master	FALSE

方向	プロトコル	ポート	namespace	Service	Pod	Container	Node Role	任意
Ingress	TCP	111	ホストシステムサービス	rpcbind			master	TRUE
Ingress	TCP	2379	openshift-etcd	etcd	etcd	etcdctl	master	FALSE
Ingress	TCP	2380	openshift-etcd	healthz	etcd	etcd	master	FALSE
Ingress	TCP	5050	openshift-machine-api		ironic-proxy	ironic-proxy	master	FALSE
Ingress	TCP	6080	openshift-kube-apiserver		kube-apiserver	kube-apiserver-insecure-readyz	master	FALSE
Ingress	TCP	6385	openshift-machine-api		ironic-proxy	ironic-proxy	master	FALSE
Ingress	TCP	6443	openshift-kube-apiserver	apiserver	kube-apiserver	kube-apiserver	master	FALSE
Ingress	TCP	8080	openshift-network-operator		network-operator	network-operator	master	FALSE
Ingress	TCP	8798	openshift-machine-config-operator	machine-config-daemon	machine-config-daemon	machine-config-daemon	master	FALSE
Ingress	TCP	9001	openshift-machine-config-operator	machine-config-daemon	machine-config-daemon	kube-rbac-proxy	master	FALSE

方向	プロトコル	ポート	namespace	Service	Pod	Container	Node Role	任意
Ingress	TCP	9099	openshift-cluster-version	cluster-version-operator	cluster-version-operator	cluster-version-operator	master	FALSE
Ingress	TCP	9100	openshift-monitoring	node-exporter	node-exporter	kube-rbac-proxy	master	FALSE
Ingress	TCP	9103	openshift-ovn-kubernetes	ovn-kubernetes-node	ovnkube-node	kube-rbac-proxy-node	master	FALSE
Ingress	TCP	9104	openshift-network-operator	metrics	network-operator	network-operator	master	FALSE
Ingress	TCP	9105	openshift-ovn-kubernetes	ovn-kubernetes-node	ovnkube-node	kube-rbac-proxy-ovn-metrics	master	FALSE
Ingress	TCP	9107	openshift-ovn-kubernetes	egressip-node-healthcheck	ovnkube-node	ovnkube-controller	master	FALSE
Ingress	TCP	9108	openshift-ovn-kubernetes	ovn-kubernetes-control-plane	ovnkube-control-plane	kube-rbac-proxy	master	FALSE
Ingress	TCP	9192	openshift-cluster-machine-approver	machine-approver	machine-approver	kube-rbac-proxy	master	FALSE

方向	プロトコル	ポート	namespace	Service	Pod	Container	Node Role	任意
Ingress	TCP	9258	openshift-cloud-controller-manager-operator	machine-approver	cluster-cloud-controller-manager	cluster-cloud-controller-manager	master	FALSE
Ingress	TCP	9444	openshift-kni-infra		haproxy	haproxy	master	FALSE
Ingress	TCP	9445	openshift-kni-infra		haproxy	haproxy	master	FALSE
Ingress	TCP	9447	openshift-machine-api		metal3-baremetal-operator		master	FALSE
Ingress	TCP	9537	ホストシステムサービス	crio-metrics			master	FALSE
Ingress	TCP	9637	openshift-machine-config-operator	kube-rbac-proxy-crio	kube-rbac-proxy-crio	kube-rbac-proxy-crio	master	FALSE
Ingress	TCP	9978	openshift-etcd	etcd	etcd	etcd メトリック	master	FALSE
Ingress	TCP	9979	openshift-etcd	etcd	etcd	etcd メトリック	master	FALSE
Ingress	TCP	9980	openshift-etcd	etcd	etcd	etcd	master	FALSE
Ingress	TCP	10250	ホストシステムサービス	kubelet			master	FALSE

方向	プロトコル	ポート	namespace	Service	Pod	Container	Node Role	任意
Ingress	TCP	10256	openshift-ovn-kubernetes	ovnkube	ovnkube	ovnkube-controller	master	FALSE
Ingress	TCP	10257	openshift-kube-controller-manager	kube-controller-manager	kube-controller-manager	kube-controller-manager	master	FALSE
Ingress	TCP	10258	openshift-cloud-controller-manager-operator	cloud-controller	cloud-controller-manager	cloud-controller-manager	master	FALSE
Ingress	TCP	10259	openshift-kube-scheduler	scheduler	openshift-kube-scheduler	kube-scheduler	master	FALSE
Ingress	TCP	10260	openshift-cloud-controller-manager-operator	cloud-controller	cloud-controller-manager	cloud-controller-manager	master	FALSE
Ingress	TCP	10300	openshift-cluster-csi-drivers	csi-liveness probe	csi-driver-node	csi-driver	master	FALSE
Ingress	TCP	10309	openshift-cluster-csi-drivers	csi-node-driver	csi-driver-node	csi-node-driver-registrar	master	FALSE
Ingress	TCP	10357	openshift-kube-apiserver	openshift-kube-apiserver-healthz	kube-apiserver	kube-apiserver-check-endpoints	master	FALSE

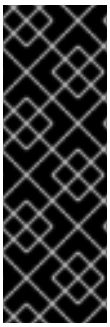
方向	プロトコル	ポート	namespace	Service	Pod	Container	Node Role	任意
Ingress	TCP	17697	openshift-kube-apiserver	openshift-kube-apiserver-healthz	kube-apiserver	kube-apiserver-check-endpoints	master	FALSE
Ingress	TCP	18080	openshift-kni-infra		CoreDNS	CoreDNS	master	FALSE
Ingress	TCP	22623	openshift-machine-config-operator	machine-config-server	machine-config-server	machine-config-server	master	FALSE
Ingress	TCP	22624	openshift-machine-config-operator	machine-config-server	machine-config-server	machine-config-server	master	FALSE
Ingress	UDP	53	openshift-dns	dns-default	dnf-default	dns	master	FALSE
Ingress	UDP	111	ホストシステムサービス	rpcbind			master	TRUE
Ingress	UDP	6081	openshift-ovn-kubernetes	ovn-kubernetes geneve			master	FALSE
Ingress	TCP	22	ホストシステムサービス	sshd			worker	TRUE
Ingress	TCP	53	openshift-dns	dns-default	dnf-default	dns	worker	FALSE
Ingress	TCP	80	openshift-ingress	router-default	router-default	ルーター	worker	FALSE
Ingress	TCP	111	ホストシステムサービス	rpcbind			worker	TRUE

方向	プロトコル	ポート	namespace	Service	Pod	Container	Node Role	任意
Ingress	TCP	443	openshift-ingress	router-default	router-default	ルーター	worker	FALSE
Ingress	TCP	8798	openshift-machine-config-operator	machine-config-daemon	machine-config-daemon	machine-config-daemon	worker	FALSE
Ingress	TCP	9001	openshift-machine-config-operator	machine-config-daemon	machine-config-daemon	kube-rbac-proxy	worker	FALSE
Ingress	TCP	9100	openshift-monitoring	node-exporter	node-exporter	kube-rbac-proxy	worker	FALSE
Ingress	TCP	9103	openshift-ovn-kubernetes	ovn-kubernetes-node	ovnkube-node	kube-rbac-proxy-node	worker	FALSE
Ingress	TCP	9105	openshift-ovn-kubernetes	ovn-kubernetes-node	ovnkube-node	kube-rbac-proxy-ovn-metrics	worker	FALSE
Ingress	TCP	9107	openshift-ovn-kubernetes	egressip-node-healthcheck	ovnkube-node	ovnkube-controller	worker	FALSE
Ingress	TCP	9537	ホストシステムサービス	crio-metrics			worker	FALSE
Ingress	TCP	9637	openshift-machine-config-operator	kube-rbac-proxy-crio	kube-rbac-proxy-crio	kube-rbac-proxy-crio	worker	FALSE

方向	プロトコル	ポート	namespace	Service	Pod	Container	Node Role	任意
Ingress	TCP	10250	ホストシステムサービス	kubelet			worker	FALSE
Ingress	TCP	10256	openshift-ovn-kubernetes	ovnkube	ovnkube	ovnkube-controller	worker	TRUE
Ingress	TCP	10300	openshift-cluster-csi-drivers	csi-liveness probe	csi-driver-node	csi-driver	worker	FALSE
Ingress	TCP	10309	openshift-cluster-csi-drivers	csi-node-driver-registrar	csi-driver-node	csi-node-driver-registrar	worker	FALSE
Ingress	TCP	18080	openshift-kni-infra		CoreDNS	CoreDNS	worker	FALSE
Ingress	UDP	53	openshift-dns	dns-default	dnf-default	dns	worker	FALSE
Ingress	UDP	111	ホストシステムサービス	rpcbind			worker	TRUE
Ingress	UDP	6081	openshift-ovn-kubernetes	ovn-kubernetes geneve			worker	FALSE

25.3. LINUX CONTROL GROUP バージョン 1 (CGROUP V1) の有効化

OpenShift Container Platform 4.14 以降、OpenShift Container Platform はクラスター内で Linux コントロールグループバージョン 2 (cgroup v2) を使用します。OpenShift Container Platform 4.13 以前で cgroup v1 を使用している場合、OpenShift Container Platform 4.15 に移行しても、cgroup 設定はバージョン 2 に自動的に更新されません。OpenShift Container Platform 4.14 以降の新規インストールでは、デフォルトで cgroup v2 が使用されます。ただし、インストール時に Linux コントロールグループバージョン 1 (cgroup v1) を有効にできます。OpenShift Container Platform で cgroup v1 を有効にすると、クラスター内のすべての cgroup v2 コントローラーと階層が無効になります。



重要

cgroup v1 は非推奨の機能です。非推奨の機能は依然として OpenShift Container Platform に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

OpenShift Container Platform で非推奨となったか、削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノートの **非推奨および削除された機能**セクションを参照してください。

cgroup v2 は、Linux cgroup API の現行バージョンです。cgroup v2 では、統一された階層、安全なサブツリー委譲、Pressure Stall Information 等の新機能、および強化されたリソース管理および分離など、cgroup v1 に対していくつかの改善が行われています。ただし、cgroup v2 には、cgroup v1 とは異なる CPU、メモリー、および I/O 管理特性があります。したがって、一部のワークロードでは、cgroup v2 を実行するクラスター上のメモリーまたは CPU 使用率にわずかな違いが発生する可能性があります。

必要に応じて、**node.config** オブジェクトを編集することで、cgroup v1 と cgroup v2 を切り替えることができます。詳細については、このセクションの「その他のリソース」の「ノードでの Linux cgroup の設定」を参照してください。

25.3.1. インストール時の Linux cgroup v1 の有効化

インストールマニフェストを作成して、クラスターのインストール時に Linux Control Group バージョン 1 (cgroup v1) を有効化できます。

重要

cgroup v1 は非推奨の機能です。非推奨の機能は依然として OpenShift Container Platform に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

OpenShift Container Platform で非推奨となったか、削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノートの **非推奨および削除された機能**セクションを参照してください。

手順

1. **node.config** オブジェクトを作成または編集して、**v1** cgroup を指定します。

```
apiVersion: config.openshift.io/v1
kind: Node
metadata:
  name: cluster
spec:
  cgroupMode: "v2"
```

2. 通常通りにインストールを続行します。

関連情報

- OpenShift Container Platform インストールの概要
- ノードでの Linux cgroup の設定

第26章 インストールの検証

インストール後に、本書の手順を実行して OpenShift Container Platform クラスターのステータスを確認できます。

26.1. インストールログの確認

OpenShift Container Platform インストールログでインストールの概要を確認できます。インストールに成功すると、クラスターへのアクセスに必要な情報はログに追加されます。

前提条件

- インストールホストにアクセスできる。

手順

- インストールホストのインストールディレクトリーにある **.openshift_install.log** ログファイルを確認します。

```
$ cat <install_dir>/.openshift_install.log
```

出力例

以下の例で説明されているように、インストールに成功すると、クラスター認証情報はログの末尾に追加されます。

```
...
time="2020-12-03T09:50:47Z" level=info msg="Install complete!"
time="2020-12-03T09:50:47Z" level=info msg="To access the cluster as the system:admin
user when using 'oc', run 'export KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'"
time="2020-12-03T09:50:47Z" level=info msg="Access the OpenShift web-console here:
https://console-openshift-console.apps.mycluster.example.com"
time="2020-12-03T09:50:47Z" level=info msg="Login to the console with user: \"kubeadmin\",
and password: \"password\""
time="2020-12-03T09:50:47Z" level=debug msg="Time elapsed per stage:"
time="2020-12-03T09:50:47Z" level=debug msg="    Infrastructure: 6m45s"
time="2020-12-03T09:50:47Z" level=debug msg="Bootstrap Complete: 11m30s"
time="2020-12-03T09:50:47Z" level=debug msg=" Bootstrap Destroy: 1m5s"
time="2020-12-03T09:50:47Z" level=debug msg=" Cluster Operators: 17m31s"
time="2020-12-03T09:50:47Z" level=info msg="Time elapsed: 37m26s"
```

26.2. イメージのプルソースの表示

ネットワークに制限のないクラスターの場合には、**crictl images** など、ノードでコマンドを使用して、プルしたイメージのソースを表示できます。

ただし、非接続インストールでは、プルされたイメージのソースを表示するには、以下の手順のように CRI-O ログを確認して、**Trying to access** のログエントリーを特定する必要があります。**crictl images** コマンドなど、イメージプルソースを表示する他の方法では、イメージがミラーリングされた場所からプルされている場合でも、ミラーリングされていないイメージ名を表示します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- マスターまたはワーカーノードの CRI-O ログを確認します。

```
$  oc adm node-logs <node_name> -u crio
```

出力例

Trying to access ログエントリーは、イメージがプルされる場所を示します。

```
...
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1366]: time="2021-08-05
10:33:21.594930907Z" level=info msg="Pulling image: quay.io/openshift-release-dev/ocp-
release:4.10.0-ppc64le" id=abcd713b-d0e1-4844-ac1c-474c5b60c07c
name=/runtime.v1alpha2.ImageService/PullImage
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1484]: time="2021-03-17
02:52:50.194341109Z" level=info msg="Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\""
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1484]: time="2021-03-17
02:52:50.226788351Z" level=info msg="Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\""
...
```

ログは、前述の例のように、イメージのプルソースを 2 回表示する場合があります。

ImageContentSourcePolicy オブジェクトに複数のミラーをリスト表示する場合には、OpenShift Container Platform はイメージを設定にリスト表示されている順序でプルしようとします。以下に例を示します。

```
Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\"
Trying to access \"li0317gcp2.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\"
```

26.3. クラスターのバージョン、ステータス、および更新の詳細の取得

oc get clusterversion コマンドを実行して、クラスターのバージョンおよびステータスを表示できます。ステータスがインストールが進行中であることを示す場合、Operator のステータスで詳細を確認できます。

現在の更新チャネルをリスト表示し、利用可能なクラスターの更新を確認することもできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

手順

1. クラスターのバージョンと全体のステータスを取得します。

```
$ oc get clusterversion
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.6.4     True        False         6m25s   Cluster version is 4.6.4
```

この出力例は、クラスターが正常にインストールされていることを示しています。

2. クラスターのステータスがインストールが進行中であることを示す場合、Operator のステータスを確認してより詳細な進捗情報を取得できます。

```
$ oc get clusteroperators.config.openshift.io
```

3. クラスター仕様、更新の可用性、および更新履歴の詳細な要約を取得します。

```
$ oc describe clusterversion
```

4. 現在の更新チャネルをリスト表示します。

```
$ oc get clusterversion -o jsonpath='{.items[0].spec}{"\n"}'
```

出力例

```
{"channel":"stable-4.6","clusterID":"245539c1-72a3-41aa-9cec-72ed8cf25c5c"}
```

5. 利用可能なクラスターの更新を確認します。

```
$ oc adm upgrade
```

出力例

```
Cluster version is 4.6.4

Updates:

VERSION IMAGE
4.6.6   quay.io/openshift-release-dev/ocp-release@sha256:c7e8f18e8116356701bd23ae3a23fb9892dd5ea66c8300662ef30563d7104f39
```

関連情報

- インストールの進行中に Operator ステータスをクエリーする方法の詳細は、インストール後に Operator ステータスをクエリーする を参照してください。
- Operator の問題の調査については、Operator の問題のトラブルシューティング を参照してください。

- クラスターの更新に関する詳細は、Web コンソールを使用したクラスターの更新 を参照してください。
- 更新リリースチャネルの詳細は、更新チャネルとリリースについて を参照してください。

26.4. クラスターが短期認証情報を使用していることの確認

Cloud Credential Operator (CCO)設定およびクラスター内の他の値をチェックして、クラスターが個々のコンポーネントに短期間のセキュリティー認証情報を使用していることを確認できます。

前提条件

- Cloud Credential Operator ユーティリティー (**ccoctl**) を使用して OpenShift Container Platform クラスターをデプロイし、短期認証情報を実装しました。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** 権限を持つユーザーとしてログインしている。

手順

- 以下のコマンドを実行して、CCO が手動モードで動作するよう設定されていることを確認します。

```
$ oc get cloudcredentials cluster \
  -o=jsonpath={.spec.credentialsMode}
```

以下の出力は、CCO が手動モードで動作していることを確認します。

出力例

```
Manual
```

- 次のコマンドを実行して、クラスターに **root** 認証情報がないことを確認します。

```
$ oc get secrets \
  -n kube-system <secret_name>
```

<secret_name> は、クラウドプロバイダーのルートシークレットの名前です。

プラットフォーム	Secret 名
Amazon Web Services (AWS)	**aws-creds**
Microsoft Azure	**azure-credentials**
Google Cloud Platform (GCP)	**gcp-credentials**

エラーは、ルートシークレットがクラスター上に存在しないことを確認します。

AWS クラスターの出力例

```
Error from server (NotFound): secrets "aws-creds" not found
```

- 次のコマンドを実行して、コンポーネントが個々のコンポーネントに対して短期セキュリティー認証情報を使用していることを確認します。

```
$ oc get authentication cluster \
  -o jsonpath \
  --template='{ .spec.serviceAccountIssuer }'
```

このコマンドは、クラスター **Authentication** オブジェクトの **.spec.serviceAccountIssuer** パラメーターの値を表示します。クラウドプロバイダーに関連付けられた URL の出力は、クラスターがクラスターの外部から作成および管理される短期認証情報を使用して手動モードを使用していることを示します。

- Azure clusters: 次のコマンドを実行して、シークレットマニフェストで指定されている Azure クライアント ID をコンポーネントが想定していることを確認します。

```
$ oc get secrets \
  -n openshift-image-registry installer-cloud-credentials \
  -o jsonpath='{.data}'
```

azure_client_id および **azure_federated_token_file** を含む出力は、コンポーネントが Azure クライアント ID を想定していることを確認します。

- Azure clusters: 次のコマンドを実行して、Pod ID Webhook が実行されていることを確認します。

```
$ oc get pods \
  -n openshift-cloud-credential-operator
```

出力例

```
NAME                                        READY   STATUS    RESTARTS   AGE
cloud-credential-operator-59cf744f78-r8pbq   2/2     Running   2          71m
pod-identity-webhook-548f977b4c-859lz        1/1     Running   1          70m
```

26.5. CLI を使用したクラスターノードのステータスのクエリー

インストール後にクラスターノードのステータスを確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. クラスターノードのステータスをリスト表示します。出力に、予想されるすべてのコントロールプレーンおよびコンピュートノードのリストが表示され、各ノードのステータスが **Ready** であることを確認します。

```
$ oc get nodes
```

出力例

```
NAME                         STATUS   ROLES    AGE   VERSION
compute-1.example.com        Ready    worker   33m   v1.29.4
control-plane-1.example.com  Ready    master   41m   v1.29.4
control-plane-2.example.com  Ready    master   45m   v1.29.4
compute-2.example.com        Ready    worker   38m   v1.29.4
compute-3.example.com        Ready    worker   33m   v1.29.4
control-plane-3.example.com  Ready    master   41m   v1.29.4
```

2. 各クラスターノードの CPU およびメモリーリソースの可用性を確認します。

```
$ oc adm top nodes
```

出力例

```
NAME                         CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
compute-1.example.com        128m         8%     1132Mi          16%
control-plane-1.example.com  801m         22%    3471Mi          23%
control-plane-2.example.com  1718m        49%    6085Mi          40%
compute-2.example.com        935m         62%    5178Mi          75%
compute-3.example.com        111m         7%     1131Mi          16%
control-plane-3.example.com  942m         26%    4100Mi          27%
```

関連情報

- ノードの健全性確認とノード問題の調査方法に関する詳細は、ノードの健全性の確認 を参照してください。

26.6. OPENSHIFT CONTAINER PLATFORM WEB コンソールでのクラスターステータスの確認

以下の情報は、OpenShift Container Platform Web コンソールの **Overview** ページで確認できます。

- クラスターの一般的なステータス
- コントロールプレーン、クラスター Operator、およびストレージのステータス
- CPU、メモリー、ファイルシステム、ネットワーク転送、および Pod の可用性
- クラスターの API アドレス、クラスター ID、およびプロバイダーの名前
- クラスターのバージョン情報
- 現在の更新チャネルの詳細や利用可能な更新を含むクラスター更新のステータス
- ノード、Pod、ストレージクラスの詳細を示すクラスターインベントリー、および永続ボリューム要求 (PVC) 情報
- 継続中のクラスターのアクティビティーおよび最近のイベントのリスト

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **Administrator** パースペクティブで、**Home** → **Overview** に移動します。

26.7. RED HAT OPENSHIFT CLUSTER MANAGER のクラスターステータスの確認

OpenShift Container Platform Web コンソールから、OpenShift Cluster Manager でクラスターのステータスに関する詳細情報を確認できます。

前提条件

- OpenShift Cluster Manager にログインしている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. OpenShift Cluster Manager の **Clusters** リストに移動し、OpenShift Container Platform クラスターを見つけます。
2. クラスターの **Overview** タブをクリックします。
3. クラスターに関する以下の情報を確認します。
   - vCPU およびメモリー可用性およびリソースの使用状況
   - クラスター ID、ステータス、タイプ、リージョン、およびプロバイダー名
   - ノード数 (ノードタイプ別)
   - クラスターバージョンの詳細、クラスターの作成日、およびクラスター所有者の名前
   - クラスターのライフサイクルサポートのステータス
   - サービスレベルアグリーメント (SLA) のステータス、サブスクリプションユニットタイプ、クラスターの実稼働ステータス、サブスクリプションの義務、サービスレベルなどのサブスクリプション情報

     ヒント

     クラスターの履歴を表示するには、**Cluster history** タブをクリックします。

4. **Monitoring** ページに移動し、以下の情報を確認します。
   - 検出されたすべての問題のリスト
   - 実行されるアラートのリスト
   - クラスター Operator のステータスおよびバージョン
   - クラスターリソースの使用状況
5. オプション: **Overview** メニューに移動して、Red Hat Insights が収集するクラスターに関する情報を表示できます。このメニューから、次の情報を表示できます。

- リスクのレベルで分類された、クラスターがさらされる可能性のある問題
- カテゴリー別のヘルスチェックのステータス

関連情報

- クラスターの潜在的な問題を特定する方法の詳細は、Insights の使用によるクラスター関連の問題の特定 を参照してください。

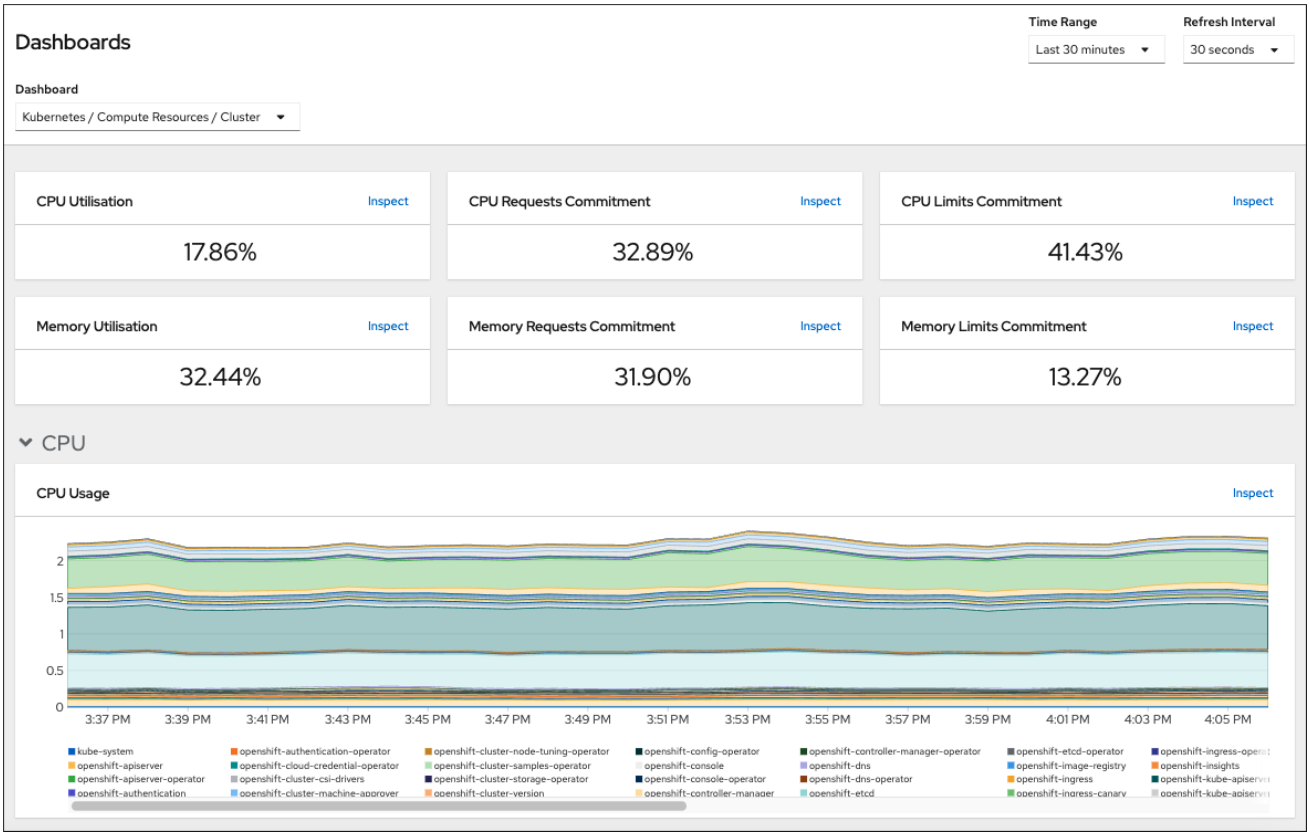
26.8. クラスターリソースの可用性および使用状況の確認

OpenShift Container Platform は、クラスターコンポーネントの状態を理解するのに役立つ包括的なモニタリングダッシュボードのセットを提供します。

Administrator パースペクティブでは、以下を含む OpenShift Container Platform のコアコンポーネントのダッシュボードにアクセスできます。

- etcd
- Kubernetes コンピュートリソース
- Kubernetes ネットワークリソース
- Prometheus
- クラスターおよびノードのパフォーマンスに関連するダッシュボード

図26.1 コンピュートリソースダッシュボードの例



前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. OpenShift Container Platform Web コンソールの **Administrator** パースペクティブで、**Observe → Dashboards** に移動します。
2. **Dashboard** リストでダッシュボードを選択します。**etcd** ダッシュボードなどの一部のダッシュボードは、選択時に追加のサブメニューを生成します。
3. 必要に応じて、**Time Range** リストでグラフの時間範囲を選択します。
   - 事前定義済みの期間を選択します。
   - **時間範囲** リストで **カスタムの時間範囲** を選択して、カスタムの時間範囲を設定します。
     a. **From** および **To** の日付と時間を入力または選択します。
     b. **Save** をクリックして、カスタムの時間範囲を保存します。
4. オプション: **Refresh Interval** を選択します。
5. 特定の項目についての詳細情報を表示するには、ダッシュボードの各グラフにカーソルを合わせます。

関連情報

- OpenShift Container Platform モニタリングスタックの詳細は、Monitoring Overviewを参照してください。

26.9. 実行されるアラートのリスト表示

アラートは、定義された条件のセットが OpenShift Container Platform クラスターで true の場合に通知を提供します。OpenShift Container Platform Web コンソールでアラート UI を使用して、クラスターで実行されているアラートを確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **Administrator** パースペクティブで、**Observe → Alerting → Alerts** ページに移動します。
2. **Severity**、 **State**、および **Source** が含まれる、実行されているアラートを確認します。
3. **Alert Details** ページで詳細情報を表示するためにアラートを選択します。

関連情報

- OpenShift Container Platform のアラートの詳細は、アラートの管理 を参照してください。

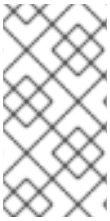
26.10. 次のステップ

- クラスターのインストール時に問題が発生した場合は、インストールのトラブルシューティング を参照してください。

- OpenShift Container Platform のインストール後に、クラスターをさらに拡張し、カスタマイズ できます。

第27章 インストールの問題のトラブルシューティング

失敗した OpenShift Container Platform インストールのトラブルシューティングを支援するために、ブートストラップおよびコントロールプレーンマシンからログを収集できます。インストールプログラムからデバッグ情報を取得することもできます。ログとデバッグ情報を使用しても問題を解決できない場合は、インストールの問題が発生した場所の特定 を参照して、コンポーネント固有のトラブルシューティングを確認してください。



注記

OpenShift Container Platform のインストールが失敗し、デバッグ出力またはログにネットワークタイムアウトまたはその他の接続エラーが含まれる場合は、ファイアウォールの設定 に関するガイドラインを確認してください。ファイアウォールとロードバランサーからログを収集すると、ネットワーク関連のエラーを診断するのに役立ちます。

27.1. 前提条件

- OpenShift Container Platform クラスターのインストールを試みたが、インストールに失敗している。

27.2. 失敗したインストールのログの収集

SSH キーをインストールプログラムに指定している場合、失敗したインストールについてのデータを収集することができます。

注記

実行中のクラスターからログを収集する場合とは異なるコマンドを使用して失敗したインストールについてのログを収集します。実行中のクラスターからログを収集する必要がある場合は、**oc adm must-gather** コマンドを使用します。

前提条件

- OpenShift Container Platform のインストールがブートストラッププロセスの終了前に失敗している。ブートストラップノードは実行中であり、SSH でアクセスできる。
- **ssh-agent** プロセスはコンピューター上でアクティブであり、**ssh-agent** プロセスとインストールプログラムの両方に同じ SSH キーを提供している。
- 独自にプロビジョニングしたインフラストラクチャーにクラスターのインストールを試行した場合には、ブートストラップおよびコントロールプレーンノードの完全修飾ドメイン名がある。

手順

1. ブートストラップおよびコントロールプレーンマシンからインストールログを収集するために必要なコマンドを生成します。
   - インストーラーでプロビジョニングされたインフラストラクチャーを使用する場合は、インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

     ```
     $ ./openshift-install gather bootstrap --dir <installation_directory> 1
     ```

1 **installation_directory** は、**./openshift-install create cluster** を実行した際に指定したディレクトリーです。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムは、ホスト名または IP アドレスを指定しなくてもよいようにクラスターについての情報を保存します。

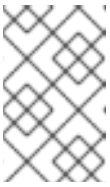
- 各自でプロビジョニングしたインフラストラクチャーを使用した場合は、インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install gather bootstrap --dir <installation_directory> \ 1
    --bootstrap <bootstrap_address> \ 2
    --master <master_1_address> \ 3
    --master <master_2_address> \ 4
    --master <master_3_address>" 5
```

1 **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

2 **<bootstrap_address>** は、クラスターのブートストラップマシンの完全修飾ドメイン名または IP アドレスです。

3 4 5 クラスター内のそれぞれのコントロールプレーン (またはマスター) マシンについては、**<master_*_address>** をその完全修飾ドメイン名または IP アドレスに置き換えます。



注記

デフォルトクラスターには 3 つのコントロールプレーンマシンが含まれます。クラスターが使用する数にかかわらず、表示されるようにすべてのコントロールプレーンマシンをリスト表示します。

出力例

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<installation_directory>/log-bundle-
<timestamp>.tar.gz"
```

インストールの失敗についての Red Hat サポートケースを作成する場合は、圧縮したログをケースに含めるようにしてください。

27.3. ホストへの SSH アクセスによるログの手動収集

must-gather または自動化された収集方法が機能しない場合にログを手動で収集します。

重要

デフォルトでは、OpenShift Container Platform ノードへの SSH アクセスは、Red Hat Open Stack Platform (RHOSP) ベースのインストールでは無効になっています。

前提条件

- ホストへの SSH アクセスがあること。

手順

1. 以下を実行し、**journalctl** コマンドを使用してブートストラップホストから **bootkube.service** サービスログを収集します。

```
$ journalctl -b -f -u bootkube.service
```

2. podman ログを使用して、ブートストラップホストのコンテナーログを収集します。これは、ホストからすべてのコンテナーログを取得するためにループで表示されます。

```
$ for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done
```

3. または、以下を実行し、**tail** コマンドを使用してホストのコンテナーログを収集します。

```
# tail -f /var/lib/containers/storage/overlay-containers/*/userdata/ctr.log
```

4. 以下を実行し、**journalctl** コマンドを使用して **kubelet.service** および **crio.service** サービスログをマスターホストおよびワーカーホストから収集します。

```
$ journalctl -b -f -u kubelet.service -u crio.service
```

5. 以下を実行し、**tail** コマンドを使用してマスターホストおよびワーカーホストのコンテナーログを収集します。

```
$ sudo tail -f /var/log/containers/*
```

27.4. ホストへの SSH アクセスを使用しないログの手動収集

must-gather または自動化された収集方法が機能しない場合にログを手動で収集します。

ノードへの SSH アクセスがない場合は、システムジャーナルにアクセスし、ホストで生じていることを調査できます。

前提条件

- OpenShift Container Platform のインストールが完了している。
- API サービスが機能している。
- システム管理者権限がある。

手順

1. 以下を実行し、**/var/log** の下にある **journald** ユニットログにアクセスします。

```
$ oc adm node-logs --role=master -u kubelet
```

2. 以下を実行し、**/var/log** の下にあるホストファイルのパスにアクセスします。

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

27.5. インストールプログラムからのデバッグ情報の取得

以下のアクションのいずれかを使用して、インストールプログラムからデバッグ情報を取得できます。

- 非表示の **.openshift_install.log** ファイルで過去のインストールからのデバッグ情報を確認します。たとえば、 以下を入力します。

```
$ cat ~/<installation_directory>/.openshift_install.log 1
```

1 **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。

- インストールプログラムが含まれるディレクトリーに切り替え、**--log-level=debug** でこれを再実行します。

```
$ ./openshift-install create cluster --dir <installation_directory> --log-level debug 1
```

1 **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。

27.6. OPENSHIFT CONTAINER PLATFORM クラスターの再インストール

OpenShift Container Platform のインストールに失敗して問題をデバッグおよび解決できない場合は、新しい OpenShift Container Platform クラスターのインストールを検討してください。完全に消去してから、インストールプロセスを開始しなおしてください。ユーザープロビジョニングインフラストラクチャー (UPI) をインストールする場合は、クラスターを手動で破棄し、関連するすべてのリソースを削除する必要があります。次の手順は、インストーラーでプロビジョニングされたインフラストラクチャー (IPI) のインストール用です。

手順

1. クラスターを破棄し、インストールディレクトリー内の非表示のインストーラー状態ファイルなども含めて、クラスターに関連付けられているすべてのリソースを削除します。

```
$ ./openshift-install destroy cluster --dir <installation_directory> 1
```

1 **installation_directory** は、**./openshift-install create cluster** を実行した際に指定したディレクトリーです。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

2. クラスターを再インストールする前に、インストールディレクトリーを削除してください。

```
$ rm -rf <installation_directory>
```

3. OpenShift Container Platform クラスターの新規インストール手順に従います。

関連情報

- OpenShift Container Platform クラスターのアンインストール

第28章 FIPS 暗号のサポート

OpenShift Container Platform クラスターを FIPS モードでインストールできます。

OpenShift Container Platform は FIPS 用に設計されています。FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

NIST 検証プログラムの詳細は、暗号化モジュール検証プログラム を参照してください。検証のために提出された RHEL 暗号化ライブラリーの個別バージョンの最新の NIST ステータスについては、政府の標準規格 を参照してください。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された RHEL 9 コンピューターからインストールプログラムを実行する必要があり、FIPS 対応バージョンのインストールプログラムを使用する必要があります。**Obtaining a FIPS-aware installation program using 'oc adm extract'** のセクションを参照してください。

RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

クラスター内の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの場合、この変更は、ユーザーがクラスターのデプロイメント時に変更できるクラスターオプションを制御する **install-config.yaml** ファイルのオプションのステータスに基づいてマシンがデプロイされる際に適用されます。Red Hat Enterprise Linux (RHEL) マシンでは、ワーカーマシンとして使用する予定のマシンにオペレーティングシステムをインストールする場合に FIPS モードを有効にする必要があります。

FIPS はクラスターが使用するオペレーティングシステムの初回の起動前に有効にされている必要があり、クラスターをデプロイしてから FIPS を有効にすることはできません。

28.1. OC ADM EXTRACTを使用した FIPS 対応のインストールプログラムの取得

OpenShift Container Platform では、FIPS モードでクラスターをインストールするために FIPS 対応インストールバイナリーを使用する必要があります。このバイナリーは、OpenShift CLI (**oc**)を使用してリリースイメージから抽出して取得できます。バイナリーを取得したら、クラスターのインストールに進み、**openshift-install** コマンドのすべてのインスタンスを **openshift-install-fips** に置き換えます。

前提条件

- バージョン 4.16 以降で OpenShift CLI (**oc**)をインストールしている。

手順

1. 以下のコマンドを実行して、インストールプログラムから FIPS 対応のバイナリーを抽出します。

```
$ oc adm release extract --registry-config "${pullsecret_file}" --command=openshift-install-fips --to "${extract_dir}" ${RELEASE_IMAGE}
```

ここでは、以下のようになります。

<pullsecret_file>

プルシークレットを含むファイルの名前を指定します。

<extract_dir>

バイナリーを抽出するディレクトリーを指定します。

<RELEASE_IMAGE>

使用している OpenShift Container Platform リリースの Quay.io URL を指定します。リリースイメージの検索の詳細は、**OpenShift Container Platform インストールプログラムの展開 を** 参照してください。

2. クラスターのインストールを続行し、**openshift-install** コマンドのすべてのインスタンスを **openshift-install-fips** に置き換えます。

関連情報

- OpenShift Container Platform インストールプログラムの展開

28.2. 公開 OPENSHIFT ミラーを使用した FIPS 対応のインストールプログラムの取得

OpenShift Container Platform では、FIPS モードでクラスターをインストールするために FIPS 対応インストールバイナリーを使用する必要があります。このバイナリーは、パブリック OpenShift ミラーからダウンロードすることで取得できます。バイナリーを取得したら、クラスターのインストールを続行し、**openshift-install** バイナリーのすべてのインスタンスを **openshift-install-fips** に置き換えます。

前提条件

- インターネットにアクセスできる。

手順

1. https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest-4.16/openshift-install-rhel9-amd64.tar.gz からインストールプログラムをダウンロードします。
2. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar -xvf openshift-install-rhel9-amd64.tar.gz
```

3. クラスターのインストールを続行し、**openshift-install** コマンドのすべてのインスタンスを **openshift-install-fips** に置き換えます。

28.3. OPENSHIFT CONTAINER PLATFORM での FIPS 検証

OpenShift Container Platform は、それが使用するオペレーティングシステムのコンポーネント用に RHEL および RHCOS 内の特定の FIPS 検証済みまたは進行中のモジュール (Modules in Process) モジュールを使用します。RHEL コア crypto コンポーネント たとえば、ユーザーが SSH を使用して OpenShift Container Platform クラスターおよびコンテナーに接続する場合、その接続は適切に暗号化されます。

OpenShift Container Platform コンポーネントは Go で作成され、Red Hat の golang コンパイラーを使用してビルドされます。クラスターの FIPS モードを有効にすると、暗号署名を必要とするすべての OpenShift Container Platform コンポーネントは RHEL および RHCOS 暗号ライブラリーを呼び出しま

す。

表28.1 OpenShift Container Platform 4.16 の FIPS モードの属性と制限

属性	制限事項
RHEL 8 および RHCOS オペレーティングシステムでの FIPS サポート。	FIPS 実装では、ハッシュの計算および署名の生成または検証を 1 つのステップで実行する関数を使用しません。この制限については、今後の OpenShift Container Platform リリースで継続的に評価され、改善されます。
CRI-O ランタイムの FIPS サポート。	
OpenShift Container Platform サービスの FIPS サポート。	
RHEL 8 および RHCOS バイナリーおよびイメージから取得される FIPS 検証済みまたは進行中のモジュール (Modules in Process) 暗号化モジュールおよびアルゴリズム。	
FIPS と互換性のある golang コンパイラーの使用。	TLS FIPS サポートは完全に実装されていませんが、今後の OpenShift Container Platform リリースで予定されています。
複数のアーキテクチャー間の FIPS サポート。	FIPS は現在、**x86_64**、**ppc64le**、および **s390x** アーキテクチャーを使用する OpenShift Container Platform デプロイメントでのみサポートされています。

28.4. クラスターが使用するコンポーネントでの FIPS サポート

OpenShift Container Platform クラスター自体は FIPS 検証済みまたは進行中のモジュール (Modules in Process) モジュールを使用しますが、OpenShift Container Platform クラスターをサポートするシステムが暗号化の FIPS 検証済みまたは進行中のモジュール (Modules in Process) モジュールを使用していることを確認してください。

28.4.1. etcd

etcd に保存されるシークレットが FIPS 検証済みまたは進行中のモジュール (Modules in Process) の暗号を使用できるようにするには、ノードを FIPS モードで起動します。クラスターを FIPS モードでインストールした後に、FIPS 承認の **aes cbc** 暗号アルゴリズムを使用して etcd データを暗号化 できます。

28.4.2. ストレージ

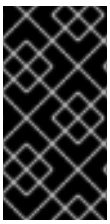
ローカルストレージの場合は、RHEL が提供するディスク暗号化または RHEL が提供するディスク暗号化を使用する Container Native Storage を使用します。RHEL が提供するディスク暗号を使用するボリュームにすべてのデータを保存し、クラスター用に FIPS モードを有効にすることで、移動しないデータと移動するデータまたはネットワークデータは FIPS の検証済みまたは進行中のモジュール (Modules in Process) の暗号化によって保護されます。ノードのカスタマイズ で説明されているように、各ノードのルートファイルシステムを暗号化するようにクラスターを設定できます。

28.4.3. ランタイム

コンテナーに対して FIPS 検証済みまたは進行中のモジュール (Modules in Process) 暗号モジュールを使用しているホストで実行されていることを認識させるには、CRI-O を使用してランタイムを管理します。

28.5. FIPS モードでのクラスターのインストール

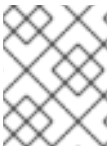
FIPS モードでクラスターをインストールするには、必要なインフラストラクチャーにカスタマイズされたクラスターをインストールする方法についての説明に従ってください。クラスターをデプロイする前に、**fips: true** を **install-config.yaml** ファイルに設定していることを確認します。



重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された RHEL コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

- Amazon Web Services
- Microsoft Azure
- ベアメタル
- Google Cloud Platform
- IBM Cloud®
- IBM Power®
- IBM Z® および IBM® LinuxONE
- RHEL KVM を使用した IBM Z® および IBM® LinuxONE
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere



注記

Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。

AES CBC 暗号化を etcd データストアに適用するには、クラスターをインストールした後に etcd データの暗号化 プロセスを実行してください。

RHEL ノードをクラスターに追加する場合は、初回の起動前に FIPS モードをマシン上で有効にしていることを確認してください。RHEL コンピュートマシンの OpenShift Container Platform クラスターへの追加 および FIPS モードでのシステムのインストール を 参照してください。